REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
09-03-2016	Ph.D. Dissertation	-

4. TITLE AND SUBTITLE
ENSEMBLE LEARNING METHOD FOR HIDDEN MARKOV MODELS

5a. CONTRACT NUMBER
W911NF-14-1-0589

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS
Anis Hamdi

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
University of Louisville
2301 S. Third Street
Jouett Hall
Louisville, KY 40208 -1838

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
66411-CS.2

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
This dissertation introduces an ensemble learning method for temporal data that uses a mixture of Hidden Markov Model (HMM) classfiers. We hypothesize that the data is generated by K models, each of which reflects a particular trend in the data. Model identfication could be achieved through clustering in the feature space or in the parameters space. However, this approach is inappropriate in the context of sequential data. The proposed approach is based on clustering in the log-likelihood space, and has two main steps. First, one HMM is fit to each of the N individual sequences. For each fitted model, we evaluate the log-likelihood of each sequence. This will result in an

15. SUBJECT TERMS
Hidden Markov Models; Ensemble methods; Landmine Detection; Ground Penetrating Radar

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Hichem Frigui
UU	UU	UU	UU		19b. TELEPHONE NUMBER: 502-852-2009

Report Title

ENSEMBLE LEARNING METHOD FOR HIDDEN MARKOV MODELS

ABSTRACT

This dissertation introduces an ensemble learning method for temporal data that uses a mixture of Hidden Markov Model (HMM) classfiers. We hypothesize that the data is generated by K models, each of which reflects a particular trend in the data. Model identfication could be achieved through clustering in the feature space or in the parameters space. However, this approach is inappropriate in the context of sequential data. The proposed approach is based on clustering in the log-likelihood space, and has two main steps. First, one HMM is fit to each of the N individual sequences. For each fitted model, we evaluate the log-likelihood of each sequence. This will result in an N-by-N log-likelihood distance matrix that will be partitioned into K groups using a relational clustering algorithm. In the second step, we pool the sequences belonging to the same cluster into K groups. Then, we learn the parameters of one HMM per group. We propose using and optimizing various training approaches for the di♂erent K groups depending on their size and homogeneity. In particular, we investigate the maximum likelihood (ML), the minimum classfication error (MCE) based discriminative, and the Variational Bayesian (VB) training approaches. Finally, to test a new sequence, its likelihood is computed in all the models and a final confidence value is assigned by combining the multiple models outputs using a decision level fusion method such as an artficial neural network or a hierarchical mixture of experts. Our approach was evaluated on two real-world applications: (1) identfication of Cardio-Pulmonary Resuscitation (CPR) scenes in video simulating medical crises; and (2) landmine detection using Ground Penetrating Radar (GPR). Results on both applications show that the proposed method can identify meaningful and coherent HMM mixture components that describe different properties of the data. Each HMM mixture component models a group of data that share common attributes. The results indicate that the proposed method outperforms the baseline HMM that uses one model for each class in the data.

ENSEMBLE LEARNING METHOD FOR HIDDEN MARKOV MODELS

By

Anis Hamdi
B.E., Signals and Systems, Polytechnic School of Tunisia, June 2002

A Dissertation
Submitted to the Faculty of the
J.B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

December 2014

ENSEMBLE LEARNING METHOD FOR HIDDEN MARKOV MODELS

By

Anis Hamdi
B.E., Signals and Systems, Polytechnic School of Tunisia, June 2002

A Dissertation Approved On

December 2, 2014

by the following Dissertation Committee:

Hichem Frigui, Ph.D., Dissertation Director

Eric C. Rouchka, Ph.D.

Roman V. Yampolskiy, Ph.D.

Tamer Inanc, Ph.D.

Wei-Bin Zeng, Ph.D.

This dissertation is dedicated to the memory of my father.

ACKNOWLEDGEMENTS

My thanks to $\cdots$

My advisor, Professor Hichem Frigui, for his valuable help, guidance, and support at all stages of this thesis.

My dissertation committee, Dr. Eric Rouchka, Dr. Roman Yampolskiy, Dr. Wei-Bin Zeng, and Dr. Tamer Inanc, for their time and consideration and insightful suggestions.

All my colleagues in the Multimedia Research Laboratory and all my friends in Louisville for their friendship and support.

Lastly but most importantly, Ghofrane, Ghaith, and all my wonderful family for the love, patience, and support they always gave me in this work and in all my projects.

ABSTRACT

ENSEMBLE LEARNING METHOD FOR HIDDEN MARKOV MODELS

Anis Hamdi

December 2, 2014

For complex classification systems, data is gathered from various sources and potentially have different representations. Thus, data may have large intra-class variations. In fact, modeling each data class with a single model might lead to poor generalization. The classification error can be more severe for temporal data where each sample is represented by a sequence of observations. Thus, there is a need for building a classification system that takes into account the variations within each class in the data.

This dissertation introduces an ensemble learning method for temporal data that uses a mixture of Hidden Markov Model (HMM) classifiers. We hypothesize that the data is generated by K models, each of which reflects a particular trend in the data. Model identification could be achieved through clustering in the feature space or in the parameters space. However, this approach is inappropriate in the context of sequential data. The proposed approach is based on clustering in the log-likelihood space, and has two main steps. First, one HMM is fit to each of the N individual sequences. For each fitted model, we evaluate the log-likelihood of each sequence. This will result in an N-by-N log-likelihood distance matrix that will be partitioned into K groups using a relational clustering algorithm. In the second step, we pool the sequences belonging to the same cluster into K groups. Then, we learn the parameters of one HMM per group. We propose using and optimizing various training approaches for the different K groups depending on their size and homogeneity. In particular, we investigate the maximum likelihood (ML), the minimum classification error (MCE) based discriminative, and the Variational Bayesian (VB) training approaches. Finally, to test a new sequence, its likelihood is computed in all the models and a final confidence value is assigned by combining the multiple models outputs using a decision level fusion method such as an artificial

neural network or a hierarchical mixture of experts.

Our approach was evaluated on two real-world applications: (1) identification of Cardio-Pulmonary Resuscitation (CPR) scenes in video simulating medical crises; and (2) landmine detection using Ground Penetrating Radar (GPR). Results on both applications show that the proposed method can identify meaningful and coherent HMM mixture components that describe different properties of the data. Each HMM mixture component models a group of data that share common attributes. The results indicate that the proposed method outperforms the baseline HMM that uses one model for each class in the data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Statistical learning problems in many fields involve sequential data. Sequential data can be temporal, i.e. ordered with respect to time, or non temporal, i.e. ordered with respect to an index other than time, e.g. space. Temporal data examples include speech signals, stock market prices, Ground Penetrating Radar (GPR) data, Cardio-Pulmonary Resuscitation (CPR) scenes, and electroencephalographic (EEG) signals. Examples of nontemporal data include pixels in an image, protein sequences, and moves in a chess game. Although there is no notion of time as such in non temporal data, it has to have an ordering, and thus can be expressed in some form of temporal data.

One of the key tasks in sequential data mining is classification, or supervised learning. Sequence classification task can be formulated as follows. Let $(\mathbf{x}_i, y_i)_{i=1}^{N}$ be a set of N training examples. Each example is a pair of sequence, $\mathbf{x}_i$, and its label, y_i. A classifier h is a function that maps from sequences to labels. In a classification setting, the goal is to learn a classifier h from the available N data samples in order to predict the label of a new example.

One of the widely used classifiers for sequential data is the Hidden Markov Model (HMM). HMMs were introduced and studied in the late 1960s and early 1970s. They have been extensively and successfully applied to speech processing and recognition by Rabiner and Huang [4]. In the 1980s, HMMs became the method of choice for sequential data in bioinformatics applications such as protein modeling [5] and gene prediction [6].

An HMM is a statistical model of a doubly stochastic process that produces a sequence of random observation vectors at discrete times according to an underlying Markov chain. The underlying Markov process, not directly observable, governs the transition of the system from one hidden state to another. At discrete instants of time, the process is assumed to be in a given state and an observation is emitted by the second stochastic process (also called observation emission probability) corresponding to the current hidden state. The underlying Markov chain then changes its state according to the system's transition matrix. In an HMM, the observer sees only the outputs of the observation emission probability and can not observe the states of the underlying

Markov chain.

In 1989, Rabiner and Huang [7] identified three key problems of interest that need to be addressed in order for HMM to be useful in real world applications. The canonical problems in HMM literature are the evaluation problem, the decoding problem, and the inference problem. One of the reasons for the wide adoption and success of HMMs is the existence of efficient and accurate algorithms that solve each of these problems. The forward-backward procedure, the Viterbi algorithm [8], and the Baum Welch reestimation algorithm [9] address respectively the aforementioned problems.

The third problem, namely the inference problem, is by far the most challenging in HMMs. It can be cast as follows: Given a set of training data, how to learn the model parameters that best fit or describe the data? The subtlety of the problem arises from the definition of the criterion that measure the model/data fit. Maximum likelihood [10], minimum classification error [11], and maximum mutual information (MMI) [12] are among such criteria that have been used in the literature.

In certain applications, the amount of data to be analyzed can be too large to be effectively handled by a single classifier. Sequential/streaming data exhibit inherently more variability as the characteristics of each class could change over time. Sometimes data are collected at a different time or in a different environment. Training a single classifier with a vast amount of heterogenous data is usually not practical. Building diverse, uncorrelated, and accurate classifiers and combining their outputs often proves to be a more efficient approach. Several ensemble learning approaches such as boosting [13], bagging [14], stacked generalization [15], random subspace method [16], and simple algebraic combiners [17] have been introduced and proven to be superior to individual classifiers in a number of benchmark datasets.

The most successful ensemble based systems (e.g. AdaBoost[13] and Bagging [14]) are typically meta-classifiers that rely on some form of resampling from the original training data and building a base classifier for each sampling outcome. Although results on benchmark datasets are significantly better than base classifiers, these methods are computationally expensive and cannot be applied efficiently to large data sets with high levels of noise and outliers. Other non-resampling-based methods [16] [17] can be applied to large datasets but are typically underperformers.

In this work, we propose an HMM-based ensemble classification method that is based on clustering sequences in the log-likelihood space. Our approach is based on partitioning the data into smaller subsets, training different HMM classifiers within the different partitions of data, and devising an adequate combination scheme to aggregate the output of the individual HMM classifiers.

We hypothesize that the data are generated by K models. These different models reflect the fact that the different classes may have different characteristics accounting for intra-class variability within the data. Model identification could be achieved through clustering in the parameters space or in the feature space. However, this approach is inappropriate as it is not trivial to define a meaningful distance metric for model parameters or sequence comparison. Our proposed approach is based on clustering in the log-likelihood space, and has two main steps. First, one HMM is fit to each of the N individual sequences. For each fitted model, we evaluate the log-likelihood of each sequence. This will result in an N x N log-likelihood distance matrix that will be partitioned into K groups using a hierarchical clustering algorithm. In the second step, we learn the parameters of one HMM per group. We propose using and optimizing various training approaches for the different K groups depending on their size and homogeneity. In particular, we will investigate the maximum likelihood, the MCE-based discriminative, and the Variational Bayesian (VB)training approaches.

The remainder of this dissertation is organized as follows. In chapter 2, we outline the background material related to HMMs and to ensemble based classification methods. In chapter 3, we detail the main components of the proposed ensemble HMM classifier. In particular, in the first part, we provide the motivations behind our approach using an illustrative example from the landmine detection application. In the second part, we detail the components of our classifier, namely the log-likelihood similarity computation, the pairwise-distance-based hierarchical clustering algorithm, the model-per-cluster learning component, and the decision level fusion approach. In chapter 4, we evaluate the proposed eHMM approach on the identification of CPR scenes in video simulating medical crises. In chapter 5, we show the experimental results of the application of our method to landmine detection using GPR data. Finally, in chapter 6, we conclude and provide potential future work.

CHAPTER 2

RELATED WORK

2.1 Hidden Markov Models

HMMs are widely used in a variety of fields for modeling time series data. Application domains of HMM include speech recognition [7], protein sequence modeling [18, 19], and finance [20, 21]. The core theory was first introduced by Baum et al in [9] with primary focus on elementary speech processing. The popularity of HMMs soared subsequently with the works of Rabiner and Huang [7, 4]. Nowadays, HMMs are the method of choice for temporal data modeling.

2.1.1 HMM architecture

An HMM is a model of doubly stochastic process that produces a sequence of random observation vectors at discrete times according to an underlying first order Markov chain. At each observation time t, the Markov chain may be in one of the N states, denoted $s_1, \cdots, s_N$. Given that the chain is at a certain state s_t, it moves to another state s_{t+1} according to a state-to-state *transition probability*, denoted $\mathbf{A}$. At any given state s_t, the model emits am observation o_t according to a state *emission probability*, denoted $\mathbf{B}$. Finally the probability that the chain starts at a particular state s_t is governed by the *initial probability* density function, denoted π.

Let T be the length of the observed sequence, $O = [o_1, \cdots, o_T]$ be the observed sequence, and $Q = [q_1, \cdots, q_T]$ be the *hidden* state sequence. The compact notation

$$\lambda = (\mathbf{A}, \mathbf{B}, \pi) \tag{1}$$

is generally used to indicate the complete parameter set of the HMM model. In (1), $\mathbf{A} = [a_{ij}]$ is the state transition probability matrix, where $a_{ij} = Pr(q_t = j | q_{t-1} = i)$ for $i, j = 1, \cdots, N$; $\mathbf{B} = b_i(o_t)$ for $i = 1, \cdots, N$ and $t = 1, \cdots, T$ where $b_i(o_t) = Pr(o_t | q_t = i)$ is the emission probability distribution in state i; and $\pi = \pi_i = Pr(q_1 = i)$ are the initial state probabilities.

An HMM is called continuous if the emission probability density functions are continuous and discrete if the emission probability density functions are discrete. In the case of a discrete HMM,

the observation vectors are commonly quantized into a finite set of M symbols $\{v_1, \cdots, v_m, \cdots, v_M\}$ called the codebook. Each state is represented by a discrete probability density function and each symbol has a probability of occurring given that the model is in that state. Thus, for the discrete case, **B** becomes a simple set of fixed probabilities for each state.

2.1.2 HMM topologies

Depending on the state transition matrix, an HMM can be classified into one of the following types [22]:

1. Ergodic model: if the associated Markov chain is ergodic. That is, the system can move from one state to any other state. In other words, all the transition matrix entries $a_{ij}, 1 \leq i, j \leq N$, are non negative.

2. Left-right model: if the Markov chain starts at a particular initial state, traverses a number of intermediate states, and finally terminates in a final state (sometimes called absorbing state). While traversing intermediate states, the chain may not go backwards to the initial state. Mathematically, the transition matrix, A, should be such that $a_{ij} = 0, \forall j < i$.

2.1.3 HMM assumptions

The theoretical framework of HMM is based on three assumptions that make the models useful in real-world applications and allow for tractable computations within the framework.

1. Markov property assumption The standard practice is to make a first order Markov assumption: The transition to a new state $q_{t+1} = s_j$, given the current and previous states, depends only on the current state $q_t = s_i$ through the transition matrix $\mathbf{A} = \mathbf{a_{ij}}$. Mathematically, this translates to: $P(q_{t+1}|q_t, q_{t-1}, \cdots, q_1, \lambda) = P(q_{t+1}|q_t, \lambda)$

2. Independence assumption Here, it is assumed that the current observation is statistically independent from all the previous observations. Formally,

$$Pr(O|Q, \lambda) = Pr(o_1, o_2, \cdots, o_T|q_1, q_2, \cdots, q_T, \lambda) = \prod_{t=1}^{T} Pr(o_t|q_t, \lambda). \tag{2}$$

3. Stationarity assumption This assumption states that the model parameters are time-independent. Particularly, for the transition matrix:

$$Pr(q_{t_1+1} = s_j|q_{t_1} = s_i, \lambda) = Pr(q_{t_2+1} = s_j|q_{t_2} = s_i, \lambda) = a_{ij}, \tag{3}$$

and for the state emission pdfs,

$$P(o_{t_1}|q_{t_1} = s_i, \lambda) = P(o_{t_2}|q_{t_2} = s_i, \lambda) \quad \sim b_i(.), \tag{4}$$

for any t_1, t_2 in $[1..T]$.

2.1.4 Main problems in HMMs

Given the form of the HMM defined in (1), Rabiner[7] defines three key problems of interest that must be solved in order for the model to be useful in real world applications:

1. The evaluation problem: given the observation sequence $\mathbf{O}$ and the model λ, how to efficiently evaluate the probability of $\mathbf{O}$ being produced by the source model λ, i.e. $Pr(\mathbf{O}|\lambda)$?

2. The decoding problem: given the observation sequence $\mathbf{O}$ and the model λ, how to find the most likely hidden state sequence $\mathbf{q}$ that led to the observed sequence $\mathbf{O}$?

3. The learning problem: also called the training problem, consists of learning the optimal model parameters given a set of training data. This problem is difficult because there are several levels of estimation required in an HMM. First, the number of states must be estimated. This is usually inferred from the physical characteristics of the problem at hand or performed using a model selection technique. Then, the model parameters $\lambda = (\pi, \mathbf{A}, \mathbf{B})$ need to be estimated. In the discrete HMM, first the codebook is determined, usually using clustering algorithms such as the K-means [23], or other vector quantization algorithms. In the continuous HMM, and for the case of Gaussian mixture density functions, the mixture component parameters, μ_{ij} and Σ_{ij}, are first initialized (usually by clustering the training data). Then for both cases, the parameters $(\pi, \mathbf{A}, \mathbf{B})$ are estimated iteratively.

2.1.5 The HMM classifier

In the previous sections, we introduced the architecture, topologies, assumptions, and the three main problems in hidden Markov modeling. The reason for the wide adoption and development of HMMs for sequential data is the presence of efficient algorithms that address and solve the aforementioned problems. HMMs have particularly proven useful in pattern recognition and classification applications. In a multiclass classification setting, each class c, $c = \{1, .., C\}$, is modeled by an HMM λ_c. An observation O is classified according to Bayes decision theory[24], which assigns O

to the class whose model has the maximum posterior probability. That is, $\hat{c} = \arg\max_c P(\lambda_c|O)$. Bayes rule states that

$$P(\lambda_c|O) = \frac{P(O|\lambda_c)P(\lambda_c)}{P(O)} \tag{5}$$

where $P(O|\lambda_c)$ is the likelihood of observed data O given the model for class c, $P(\lambda_c)$ is the prior probability of model λ_c, and $P(O)$ is the evidence or the probability of occurrence of observation O.

Assuming all the models are equally probable, and given that the denominator of equation (5) does not depend on c, maximizing the posterior probability $P(\lambda_c|O)$ is equivalent to maximizing the likelihood $P(O|\lambda_c)$: $\hat{c} = \arg\max_c P(O|\lambda_c)$. As discussed in the next section, $P(O|\lambda_c)$ can be computed efficiently using the forward-backward procedure.

A less trivial task in hidden Markov modeling is the inference, or learning of the model parameters λ_c that best describe the characteristics of class c. Several approaches to HMM model learning will be discussed later in this chapter.

2.1.6 Solution to the evaluation problem: the forward-backward procedure

Given an observation sequence $\mathbf{O} = [\mathbf{o}_1, \cdots, \mathbf{o}_T]$, the most straightforward way to compute $P(\mathbf{O}|\lambda)$ is by enumerating every possible state sequence of length T. There are N^T such states. Let $\mathbf{q} = [q_1, q_2, \cdots, q_T]$ denote one such state, where q_1 is the initial state. Assuming statistical independence of observations o_t, the probability of the observation sequence $\mathbf{O}$ given the state sequence $\mathbf{q}$ and the model λ is:

$$P(\mathbf{O}|\mathbf{q}, \lambda) = \prod_{t=1}^{T} P(\mathbf{o}_t|q_t, \lambda) = b_{q_1}(\mathbf{o}_1)b_{q_2}(\mathbf{o}_2)\cdots b_{q_T}(\mathbf{o}_T) \tag{6}$$

The probability of the fixed state sequence $\mathbf{q}$ in the model λ is:

$$P(\mathbf{q}|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T} \tag{7}$$

The joint probability of $\mathbf{O}$ and $\mathbf{q}$ is the product of the above two terms, that is:

$$P(\mathbf{O}, \mathbf{q}|\lambda) = P(\mathbf{O}|\mathbf{q}, \lambda)P(\mathbf{q}|\lambda) \tag{8}$$

To obtain the probability of $\mathbf{O}$ given the model λ, we sum the joint probability in (8) over all possible state sequences. Thus, we have

$$\begin{aligned} P(\mathbf{O}|\lambda) &= \sum_{\mathbf{q}} P(\mathbf{O}|\mathbf{q}, \lambda)P(\mathbf{q}|\lambda) \\ &= \sum_{q_1, q_2, \cdots, q_T} \pi_{q_1} b_{q_1}(\mathbf{o}_1) a_{q_1 q_2} b_{q_2}(\mathbf{o}_2) \cdots a_{q_{T-1} q_T} b_{q_T}(\mathbf{o}_T) \end{aligned} \tag{9}$$

The evaluation problem of HMMs using exhaustive enumeration is intractable as the summation in (9) is on the order of N^T computations. This evaluation is infeasible even for small values of N and T. For instance, for $N = 4$, $T = 120$, there are on the order of 10^{72} computations (on the order of the number of electrons in the universe! [25]). Fortunately, the forward-backward procedure tackles this problem and reduces the complexity to N^2T by exploiting the HMM Markov property and its interpretation as a graphical model.

2.1.6.1 Forward procedure

Let the forward variable $\alpha_t(i)$ be the probability of the partial distribution sequence up to time t:

$$\alpha_t(i) = P(\mathbf{o}_1, \mathbf{o}_2, \cdots, \mathbf{o}_t, q_t = i|\lambda). \tag{10}$$

A closed-form for $\alpha_t(i)$ can be derived by induction on t [7]:

1. Initialization

$$\alpha_1(i) = \pi_i b_i(\mathbf{o}_1), \quad 1 \leq i \leq N. \tag{11}$$

2. By induction, for $1 \leq t \leq T-1$, and $1 \leq j \leq N$, we have:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}) \tag{12}$$

3. At the end we have:

$$Pr(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{13}$$

In step 2, the forward variable $\alpha_t(j)$ is computed for each state j and each partial observation up to time t. Given a fixed t, state j can be reached independently from any of the N states with probability a_{ij}. The summation in step 2 reflects this. Once at state j, the system emits observation symbol o_t according to state j probability distribution $b_j(\cdot)$. Thus, using dynamic programming, the solution to problem 1 is reduced to the order of N^2T computations.

2.1.6.2 Backward procedure

In a similar way, we let the backward variable $\beta_t(i)$ be the probability of the partial distribution sequence from time $t+1$ to T given the state i at time t and the model λ:

$$\beta_t(i) = Pr(o_{t+1}, o_{t+2}, \ldots, o_T|q_t = i, \lambda). \tag{14}$$

Note that here $q_t = i$ has already been given (it was not the case for the forward variable). This distinction has been made to be able to combine the forward and the backward variables coherently to produce useful results that will help solve problem 2 and problem 3 of HMMs, as we shall see soon. Following the same induction argument as for $\alpha_t(i)$, $\beta_t(i)$ could be easily calculated using the following steps [7]:

1. Initialization:

$$\beta_T(i) = i, 1 \leq i \leq N_s \tag{15}$$

2. By induction and for $t = T-1, T-2, \ldots, 1$, and $1 \leq i \leq N$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \tag{16}$$

3. After termination,

$$Pr(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i) \tag{17}$$

The computation of $Pr(O|\lambda)$ using $\beta_t(i)$ also involves the order of N^2T calculations. Hence both the forward as well as the backward method are equally efficient for the computation of $Pr(O|\lambda)$. This solves the evaluation problem in HMM.

2.1.7 Solution to the decoding problem: the Viterbi algorithm

The second problem in HMM consists of finding the optimal state sequence associated with a given observation sequence. Unlike the evaluation problem, for which an exact solution could be derived, there are several possible ways for solving the decoding problem. The difficulty lies with the definition of an optimality criterion. One possible choice for the optimality criterion is to choose the states that are individually most likely at each time t. In other words, we have to find a state sequence $\mathbf{q} = [q_1, q_2, \ldots, q_T]$ such that the probability of occurrence of the observation sequence $O = [o_1, o_2, \ldots, o_T]$ from this state sequence is greater than that from any other state sequence. The problem is then to find $\mathbf{q}^*$ that will maximize $Pr(O, \mathbf{q}|\lambda)$. This can be achieved using the Viterbi algorithm [8]. First, we define the quantity

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P(q_1, q_2, \cdots, q_{t-1}, q_t = i, o_1, o_2, \cdots, o_t|\lambda) \tag{18}$$

as the highest probability along a single path that generated the partial sequence $[o_1 o_2 \cdots o_t]$ and ended in state i. By induction, we have:

$$\delta_{t+1}(j) = [\max_i \delta_t(i)a_{ij}]b_j(o_{t+1}) \tag{19}$$

The Viterbi algorithm can be summarized by the following four steps:

1. Initialization: for $1 \leq i \leq N$

$$\delta_1(i) = \pi_i b_i(o_1) \tag{20}$$

$$\psi_1(i) = 0 \tag{21}$$

2. Recursive computation: for $2 \leq t \leq T$ and $1 \leq j \leq N$

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i)a_{ij}] .b_j(o_t) \tag{22}$$

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i)a_{ij}] \tag{23}$$

3. Termination:

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{24}$$

$$q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)] \tag{25}$$

4. Tracing back the optimal state sequence, for $t = T-1, T-2, \ldots, 1$

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \tag{26}$$

Hence P^* gives the required state-optimized probability, and $\mathbf{q}^* = [q_1^*, q_2^*, \ldots, q_T^*]$ is the optimal state sequence. Computationally, the Viterbi algorithm is similar to the forward-backward procedure except for the comparisons needed to find the maximum values of δ and for the backtracking step 4. Therefore, its complexity is also on the order of N^2T. It is possible to implement the Viterbi algorithm by taking the logarithm of δ hence reducing the complexity to N^2T additions instead of multiplications.

In summary, problems 1 and 2 are efficiently solved using the forward-backward procedure and dynamic programming. However, a less trivial task in hidden Markov modeling is the inference, or learning of the model parameters λ that best describe the characteristics of the training data. In the following, we describe three different approaches to solve this problem.

2.1.8 Solutions to the inference problem

The parameter estimation problem is the most challenging problem of HMMs. The goal is to adjust the model parameters $(\pi, \mathbf{A}, \mathbf{B})$ based on the observed sequence(s) O. In fact, there is no analytical solution for the model parameter set that maximizes the probability of the observation sequence. There are several possible ways for defining an optimality criterion. The Maximum Likelihood (ML) [10] criterion is generally used to find the set of parameter estimators that maximize the likelihood of the data in the HMM model. However, for the general HMM setting, there is no analytical solution to the Maximum Likelihood Estimator (MLE). Generally, iterative methods, such us the Expectation Maximization (EM) algorithm [26], are used to approximate the MLE.

Another widely used optimality criterion is the minimum misclassification error (MCE)[11]. In the MCE training framework, the goal is to find the optimal model parameters that minimize the classification error on the training data.

Both MLE and MCE are iterative procedures and could lead to sub-optimal solutions (local optima). In addition, MLE is a point estimation of the true model parameters that maximize the posterior probability. This method requires a large training data set to get good estimates. In the case of insufficient training data, Bayesian methods are preferred. In Bayesian learning, we first set priors on the parameters to be estimated. Then, we use the available observations (evidence) to compute the posterior distribution over the parameters. Predictions for test data are made by integrating out the parameters. Unfortunately Bayesian inference for HMMs is not possible as it involves computing intractable integrals. Most existing methods, such as Markov Chain Monte Carlo (MCMC) [27], Gibbs sampling [28], and Laplace approximations[25] either require vast computational resources to get accurate results or approximate all the posteriors via a normal distribution centered at the ML estimate. The Variational Bayesian (VB)[29] method tries to approximate the posteriors by a class of simpler functions. Typically, the approximating functions should be separable/focatorizable in the model parameters.

2.1.8.1 Maximum Likelihood training: The Baum-Welch algorithm

In this section, we outline the procedure for the re-estimation of the HMM parameters using the Expectation Maximization algorithm [26]. In the context of HMMs, the EM algorithm is also known as the Baum-Welch (BW) algorithm [9], or the forward-backward reestimation procedure. The goal is to approximate the maximum likelihood estimator of the HMM parameters, since a closed-form of the these parameters can not be derived.

First, we define $\xi_t(i,j)$ as the probability of being in state i at time t and state j at time $t+1$, given the model and the observation sequence, i.e.

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j | O, \lambda). \tag{27}$$

From the definitions of the forward and backward variables in equations (10) and (14), we can rewrite $\xi_t(i,j)$ as

$$\begin{aligned} \xi_t(i,j) &= \frac{P(q_t = i, q_{t+1} = j, O|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}. \end{aligned} \tag{28}$$

Let $\gamma_t(i)$ denote the probability of being in state i at time t, given the observation O and the model λ, i.e.,

$$\gamma_t(i) = P(q_t = i | O, \lambda). \tag{29}$$

It can be shown that $\gamma_t(i)$ is related to $\xi_t(i,j)$ by summing over all states j, i.e.,

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j). \tag{30}$$

Now, if we sum $\gamma_t(i)$ over the time index t, we get a quantity that could be interpreted as the expected number of times that state i is visited. Similarly, summing $\xi_t(i,j)$ over t yields the expected number of transitions from state i to state j. $\gamma_1(i)$ is interpreted as the expected frequency (number of times) we started at state i. Finally, summing $\gamma_t(i)$ over time steps at which a particular symbol v_m was observed can be interpreted as the expected number of times visiting state i and observing v_m. Using the above interpretations, we derive the following equations for re-estimating the parameters of an HMM:

$$\overline{\pi_i} = \gamma_1(i), \tag{31}$$

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \tag{32}$$

and

$$\overline{b_j}(m) = \frac{\sum_{1\leq t\leq T, o_t=v_m} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{33}$$

Algorithm 1 Baum-Welch training algorithm

Require: Training data $[O^{(1)}, \cdots, O^{(R)}]$, $O^{(r)}=[o_1, \cdots, o_T]$. Fix the variables N and M.
Ensure:

1: Cluster training data into N clusters, and let s_i, the center of each cluster, be the representative of state i
2: Cluster training data into M clusters, and let v_m, the center of each cluster, be the m^{th} element of the codebook
3: Set the initial parameters using one of the methods described in section (2.1.9)
4: **while** stopping criteria not satisfied **do**
5: Compute the probability of the hidden states given the data using the forward-backward procedure;
6: Deduce the transition and emission statistics;
7: Update **A** using (32);
8: Update **B** using (33);
9: **end while**

A more detailed derivation of the update equations for the BW algorithm could be found in [30].

2.1.8.2 Discriminative training: Minimum Classification Error criterion

In this section, we derive the necessary MCE update equations for the HMM parameters for a C-class problem. In the discriminative MCE training, we seek to minimize the classification error. Each random sequence O is to be classified into one of the C classes. We denote these classes by $C_c, c = 1, 2, \ldots, C$. Each class c is modeled by an HMM λ_c. Let O be an observation sequence and let $g_c(O)$ be a discriminant function associated with classifier c. In the case of an HMM classifier, the discriminant function g_c is proportional to the posterior probability $Pr(O|\lambda_c)$. Thus, the sequence O is assigned to the class c in which it has maximum posterior probability.

The classifier $\Gamma(O)$ defines a mapping from the sample space $O \in \mathbb{O}$ to the discrete categorical set $C_c, c = 1, 2, \ldots, C$. That is,

$$\Gamma(O) = \arg\max_c g_c(O). \tag{34}$$

The misclassification measure of the sequence O is defined as:

$$d_c(O) = -g_c(O, \Lambda) + \log\left[\frac{1}{C-1}\sum_{j,j\neq c} \exp[\eta g_j(O, \Lambda)]\right]^{\frac{1}{\eta}} \tag{35}$$

where η is a positive number, $d_c(O) > 0$ implies misclassification and $d_c(O) \leq 0$ means correct decision. When η approaches ∞, the second term of the right hand side of equation (35) becomes $\max_{j,j\neq c} g_j(O,\Lambda)$. The misclassification measure is then casted in a smoothed zero-one function, referred to as loss function, defined as:

$$l_c(O;\Lambda) = l(d_c(O)), \tag{36}$$

where l is a sigmoid function, one example of which is:

$$l(d) = \frac{1}{1+\exp(-\gamma d+\theta)}. \tag{37}$$

In (37), the threshold parameter θ is normally set to zero, and the smoothing parameter γ is set to a number larger than one. Correct classification corresponds to loss values in $[0, \frac{1}{2})$, and misclassification corresponds to loss values in $(\frac{1}{2}, 1]$.

For an unknown sequence O, the classifier performance is measured by:

$$l(O;\Lambda) = \sum_{c=1}^{C} l_c(O;\Lambda)\mathbb{I}(O \in C_c) \tag{38}$$

where $\mathbb{I}(.)$ is the indicator function. The true classifier performance can be measured by the expected cost of the error:

$$\mathbb{L}(\Lambda) = \mathbb{E}[l(O;\Lambda)]. \tag{39}$$

Given a set of training observation sequences O_r, $r = 1, 2, \ldots, R$, an empirical loss function on the training dataset is defined as

$$\mathbf{L}(\Lambda) = \frac{1}{R}\sum_{r=1}^{R}\sum_{c=1}^{C} l_c(O_r;\Lambda)\mathbb{I}(O_r \in C_c). \tag{40}$$

The empirical loss L is a smooth function in Λ which can be optimized over the training set using gradient descent methods such as the generalized probabilistic descent algorithm (GPD) or its variants [11]. The HMM parameters are therefore derived by minimizing $\mathbf{L}(\Lambda)$ using a gradient descent algorithm.

In order to ensure that the estimated HMM parameters satisfy the stochastic constraints of $a_{ij} \geq 0$, $\sum_{j=1}^{N_s} a_{ij} = 1$ and $b_{jk} \geq 0$, $\sum_{j=1}^{M} b_{jk} = 1$, we map these parameters using

$$a_{ij} \quad \rightarrow \quad \tilde{a}_{ij} = \log a_{ij}, \tag{41}$$

$$b_{ij} \quad \rightarrow \quad \tilde{b}_{ij} = \log b_{ij} \tag{42}$$

Then, the parameters are updated with regards to $\tilde{\Lambda}$. After updating, we map them back using

$$a_{ij} = \frac{\exp \tilde{a}_{ij}}{\sum_{j'=1}^{N_s} \exp \tilde{a}_{ij'}}, \tag{43}$$

$$b_{jk} = \frac{\exp \tilde{b}_{jk}}{\sum_{k'=1}^{M} \exp \tilde{b}_{jk'}}. \tag{44}$$

Using a batch estimation mode, the HMM parameters are iteratively updated using

$$\tilde{\Lambda}(\tau+1) = \tilde{\Lambda}(\tau) - \epsilon \nabla_{\tilde{\Lambda}} \mathbf{L}(\Lambda)\Big|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}. \tag{45}$$

It can be shown [31] that the update equations for $\tilde{a}_{ij}^{(c)}$ and $\tilde{b}_{jk}^{(c)}$ are

$$\tilde{a}_{ij}^{(c)}(\tau+1) = \tilde{a}_{ij}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{a}_{ij}^{(c)}}\bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}, \tag{46}$$

and

$$\tilde{b}_{jk}^{(c)}(\tau+1) = \tilde{a}_{jk}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{jk}^{(c)}}\bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}, \tag{47}$$

where

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{a}_{ij}^{(c)}} = \sum_{r=1}^{R} \sum_{m=1}^{C} \frac{\partial l_m(O_r)}{\partial d_m(O_r)} \frac{\partial d_m(O_r)}{\partial g_c^{O_r}} \frac{\partial g_c^{O_r}}{\partial a_{ij}^{(c)}} \frac{\partial a_{ij}^{(c)}}{\partial \tilde{a}_{ij}^{(c)}},$$

and

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{ij}^{(c)}} = \sum_{r=1}^{R} \sum_{m=1}^{C} \frac{\partial l_m(O_r)}{\partial d_m(O_r)} \frac{\partial d_m(O_r)}{\partial g_c^{O_r}} \frac{\partial g_c^{O_r}}{\partial b_{ij}^{(c)}} \frac{\partial b_{ij}^{(c)}}{\partial \tilde{b}_{ij}^{(c)}}.$$

Substituting the partial derivatives, we get the gradient direction of the update equations for the parameters of the HMM:

$$\begin{aligned} \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{a}_{ij}^{(c)}} &= \sum_{r=1}^{R} \sum_{m=1}^{C} \sum_{t=1}^{T} \gamma l_m(O_r, \Lambda)(1 - l_m(O_r, \Lambda)) \times \\ &\quad \delta(q_t^r = i, q_{t+1}^r = j)(1 - a_{ij}^{(c)}) \frac{\partial d_c(O_r)}{\partial g_m(O_r, \Lambda)}, \end{aligned}$$

and

$$\begin{aligned} \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{jk}^{(c)}} &= \sum_{r=1}^{R} \sum_{m=1}^{C} \sum_{t=1}^{T} \gamma l_m(O_r, \Lambda)(1 - l_m(O_r, \Lambda)) \times \\ &\quad \delta(q_t^r = j, Q_V(o_t^r) = k)(1 - b_{jk}^{(c)}) \frac{\partial d_c(O_r)}{\partial g_m(O_r, \Lambda)}, \end{aligned}$$

where

$$\frac{\partial d_c(O)}{\partial g_m(O, \Lambda)} = \begin{cases} -1 & \text{if } c = m \\ \frac{\exp[\eta g_c(O,\Lambda)]}{\sum_{j, j \neq c} \exp[\eta g_j(O,\Lambda)]} & \text{if } c \neq m \end{cases}, \tag{48}$$

Algorithm 2 MCE/GPD training algorithm

Require: Training data $[O^{(1)}, \cdots, O^{(R)}]$, $O^{(r)}=[o_1, \cdots, o_T]$. Fix the HMM structure variables N and M for each model λ_c.

Ensure:

1: For each λ_c, cluster training data into N clusters, and let s_i, the center of each cluster, be the representative of state i.
2: For each λ_c, cluster the training data into M symbols. The center of each cluster v_m is a symbol of the codebook.
3: **while** stopping criteria not satisfied **do**
4: Compute the loss function of each sequence O using (40);
5: Update $\mathbf{A}$ of each λ_c using (46);
6: Update $\mathbf{B}$ of each λ_c using (47);
7: **end while**

and δ is the 2-D Kronecker's delta function. The MCE-based discriminative training is outlined in Algorithm 2.

A more detailed derivation of the update equations for the MCE-based discriminative training algorithm could be found in [30].

2.1.8.3 Variational Bayesian training

The Bayesian framework provides a solution to the over-fitting problem. Unfortunately, computations in the Bayesian framework can be intractable. Most existing methods, such as Markov Chain Monte Carlo (MCMC) [27] and the Laplace approximation [25] either require vast computational resources to get good estimates or they crudely approximate all the posteriors via a normal distribution. The variational Bayesian (VB) method attempts to approximate the integration as accurately as possible while remaining computationally tractable [29].

Let O , Z, Θ, and λ denote the training data set, latent (hidden) variables, a set of model parameters, and the model identity itself, respectively. In VB, the aim is to maximize the marginal likelihood defined as:

$$Pr(O|\lambda) = \int Pr(O, Z, \Theta|\lambda) dZ d\Theta. \tag{49}$$

The VB approach introduces a variational posterior distribution $\tilde{p}(Z, \Theta|\lambda)$ of the model parameters and hidden variables. Taking the logarithm, then the expectation with respect to the distribution $\tilde{p}(Z, \Theta|\lambda)$, and applying Jensen's inequality [32], we obtain:

$$\log Pr(O|\lambda) = F(\tilde{p}) + KL(\tilde{p}||Pr) \tag{50}$$

where

$$F(\tilde{p}) = \int \tilde{p}(Z, \Theta|\lambda) \log \frac{Pr(O, Z, \Theta|\lambda)}{\tilde{p}(Z, \Theta|\lambda)} dZ d\Theta, \tag{51}$$

and

$$KL(\tilde{p}||Pr) = \int \tilde{p}(Z,\Theta|\lambda)\log\frac{\tilde{p}(Z,\Theta|\lambda)}{\tilde{p}(Z,\Theta|O,\lambda)}dZd\Theta. \tag{52}$$

The term $F(\tilde{p})$ represents a lower bound of $\log Pr(O|\lambda)$. The aim of VB learning is to maximize this lower bound by tuning the variational posterior $\tilde{p}(Z,\Theta|\lambda)$ such that as the variational posterior approaches the true posterior, the bound becomes tight and the marginal likelihood can be efficiently approximated efficiently. The equality between $F(\tilde{p})$ and $\log Pr(O|\lambda)$ occurs if and only if $\tilde{p}(Z,\Theta|\lambda) = Pr(Z,\Theta|\lambda,O)$. In practice, we use a factorized approximation $\tilde{p}(Z,\Theta) \approx \tilde{p}_Z(Z)\tilde{p}_\Theta(\Theta)$. Thus, F can be written in the form:

$$F(\tilde{p}) = \int \tilde{p}_Z(Z)\tilde{p}_\Theta(\Theta)\log\frac{Pr(O,Z,\Theta|\lambda)}{\tilde{p}_Z(Z)\tilde{p}_\Theta(\Theta)}dZd\Theta = F(\tilde{p}_Z(Z),\tilde{p}_\Theta(\Theta),O,\lambda) \tag{53}$$

The quantity F can be viewed as a functional of the free distributions $\tilde{p}_Z(Z)$ and $\tilde{p}_\Theta(\Theta)$. The variational Bayesian (VB) algorithm iteratively maximizes F in equation (53) with respect to the free distributions $\tilde{p}_Z(Z)$ and $\tilde{p}_\Theta(\Theta)$. Taking the partial derivatives of F with respect to the free distributions, and setting it to zero, results in the following update equations:

$$\tilde{p}_Z^{(t+1)}(Z) \propto \exp\left[\int Pr(Z,O|\Theta,\lambda)\tilde{p}_\Theta^{(t)}(\Theta)d\Theta\right], \tag{54}$$

and

$$\tilde{p}_\Theta^{(t+1)}(\Theta) \propto Pr(\Theta|\lambda)\exp\left[\int Pr(Z,O|\Theta,\lambda)\tilde{p}_Z^{(t+1)}(Z)dZ\right] \tag{55}$$

where subscript (t) denotes the iteration number. Clearly, $\tilde{p}_{Z_i}(Z_i)$ and $\tilde{p}_\Theta(\Theta)$ are coupled, so we iterate these equations until convergence.

The variational Bayesian training steps is outlined in Algorithm 3.

Algorithm 3 VB training algorithm

Require: Training data $[O^{(1)},\cdots,O^{(R)}]$, $O^{(r)}$=$[o_1,\cdots,o_T]$. Fix the variables N and M.
Ensure:
1: Cluster training data into N clusters, and let s_i, the center of each cluster, be the representative of state i
2: Cluster training data into M clusters, and let v_m, the center of each cluster, be the k^{th} element of the codebook
3: Set priors on the HMM parameters.
4: **while** stopping criteria not satisfied **do**
5: Update marginal posteriors on the hidden variables Z using equation (54);
6: Update marginal posteriors on the parameters Θ using equation (55);
7: **end while**

A more detailed derivation of the update equations for the VB training algorithm could be found in [29].

2.1.9 Initialization of the HMM parameters

A key question in HMM is how to choose initial estimates of the HMM parameters to improve the likelihood of reaching the global minimum and the rate of convergence. A number of initialization methods have been proposed [7]. Examples include:

- Random: the HMM parameters are generated randomly from a uniform distribution.
- Manual segmentation: when the hidden states have a physical meaning, manual partitioning could be performed to split the data into the different states of the HMM and then the remaining parameters could be derived [33].
- Segmental k-means: starting form a random guess of the HMM parameters, and using the Viterbi algorithm to label the observation sequences, the segmental k-means clusters the sequences to learn the HMM parameters [7].

2.2 MultiStream HMMs

For complex classification, multiple sources of information may contribute to the generation of sequences. In the standard HMMs, the different features contribute equally to the classification decision. However, these features may have different relevance degrees that depend on different regions of the feature space. Thus, more complex HMM based structures were proposed [34, 30] to handle temporal data with multiple sources of information. Approaches toward the combination of different modalities can be divided into three main categories:

- Feature level fusion: multiple features are concatenated into a large feature vector and a single HMM model is trained [35]. This type of fusion has the drawback of treating heterogeneous features equally important. It also cannot represent the loose timing synchronicity between different modalities easily.
- Decision level fusion: the modalities are processed separately to build independent models [36]. This approach completely ignores the correlation between features and allows complete asynchrony between the streams.
- Model level fusion: an HMM model that is more complex than a standard one is sought. This additional complexity is needed to handle the correlation between modalities, and the loose synchronicity between sequences.

Several HMM structures have been proposed in the context of model level fusion. Examples include factorial HMM [37], coupled HMM [38] and Multi-stream HMM [34]. Both factorial and coupled HMM structures allow asynchrony between sequences since a separate state sequence is assigned to each stream [39]. However, this is performed at the expense of an approximate parameter estimation. In fact, the parameters of factorial and coupled HMMs could be estimated via the EM (Baum-Welch) algorithm [26]. However, the E- step is computationally intractable and approximation approaches are used instead [38, 37].

Multi-Stream HMM (MSHMM) [34, 30] is an HMM based structure that handles multiple modalities for temporal data. In [30], weights are assigned to the different streams and incorporated into the model parameters. The added complexity of model parameters makes the inference task more challenging and in some cases, even the ML training becomes intractable. In [30], multiple MSHMM structures for the discrete and continuous HMM have been proposed. Generalized training framework combining the ML and the MCE training were derived to learn the MSHMM parameters simultaneously.

MSHMM addresses the issue of fusing multiple features. Another problem in complex real-world applications is that a single model is not sufficient to accurately handle the intra-class variability. To alleviate this limitation, multiple simple models could be used and fused. The models need not to be accurate on all the regions of the feature space. The multiple model approach can be regarded as ensemble learning. Generally, an ensemble learning method includes two main steps that could potentially be intertwined: the creation and the fusion of the multiple models. Different approaches of ensemble learning are described in the following section.

2.3 Ensemble learning methods

In this section, we outline some ensemble learning approaches that have been extensively studied and used in many applications.

2.3.1 Bagging

Bagging is considered as a method of independent construction of the base classifiers [14]. It consists of running a learning algorithm several times with random samples from the training data for each run. Given a set of N training data points, at each iteration, bagging chooses a set of data points of size $m \leq N$ via uniform sampling with replacement from the original data points. At each run, some data points will appear multiple times, other data points will not appear at all in the

resampled dataset. If the base classifier is unstable, i.e. a small variation in the training data leads to large change in the resulting classifier, the bagging will result in a diverse set of hypotheses. On the other hand, if the base classifier is stable, Bagging may underperform the base classifier.

2.3.2 Boosting (AdaBoost)

Following the taxonomy in [40], Boosting is a method of coordinated construction of ensembles. While in bagging the samples are drawn uniformly with replacement, in boosting methods the learning algorithm is called at each iteration using a different distribution over the training examples. At iteration $t+1$, this technique places higher weights on the examples misclassified by the base classifier at iteration t. Let $\{\mathbf{x}_i, y_i)_{i=1}^N, y_i = \pm 1\}$ be a set of N labeled training examples. The goal is to construct a weighted sum of hypothesis such that $H(\mathbf{x}_i) = \sum_t w_t h_t(\mathbf{x}_i)$ has the same sign as y_i. The algorithm operates as follows. Let $d_t(\mathbf{x}_i)$ be the weight of data point $\mathbf{x}_i$ at iteration t of the algorithm. Initially, all training data points $\mathbf{x}_i$ are given a uniform weight ($d_1(\mathbf{x}_i) = 1/N$). At iteration t, the underlying learning algorithm constructs hypotheses h_t to minimize the weighted training error. The resulting weighted error on the training set is $r_t = \sum_t w_t d_t(\mathbf{x}_i) h_t(\mathbf{x}_i)$. The weight assigned to hypothesis h_t is

$$w_t = \frac{1}{2} ln(\frac{(1+r_t)}{(1-r_t)}). \tag{56}$$

For each data point i, the weight for the next iteration is updated using

$$d_{t+1}(\mathbf{x}_i) = d_t(\mathbf{x}_i) \frac{\exp(-w_t y_i h_t(\mathbf{x}_i))}{Z_t}, \tag{57}$$

where Z_t is a normalizing factor.

It has been shown in [41] that this algorithm is a form of gradient optimization in the function space with $J(H) = \sum_i exp(-y_i H(\mathbf{x}_i))$ as the objective function. The quantity $y_i H(\mathbf{x}_i)$ could be interpreted as the margin by which $\mathbf{x}_i$ is correctly classified. Minimizing J is equivalent to maximizing the margin. Work by Vapnik [42] on SVMs (Support Vector Machines) derives upper bounds on the generalization error with regards to the margins of the training data. Freund et al [43] extended the margin analysis to Adaboost. The generalization error is bounded by the fraction of training data for which the margin is less than a quantity $\Theta > 0$, plus a term that grows as

$$\sqrt{\frac{d}{N}} \frac{ln(\frac{N}{d})}{\Theta}, \tag{58}$$

where d is a measure of the expressive power of the hypothesis space from which the individual classifiers are drawn, known as the VC-dimension. The value of Θ can be optimized such that the

bound in (58) is minimized. Experimentally Adaboost has been proven to increase the margins on the training data points which explains partly its good generalization performance on new data points.

Although Adaboost shows significant improvement over existing classification methods on benchmark datasets, it suffers from the following drawbacks:

- Adaboost does not perform well on high noise setting as it tends to put very high weights on the noisy data points
- The upper bounds on the generalization error are not tight and can not fully explain the success of Adaboost across multiple benchmark datasets. Dietterich [44] noticed that it is possible to design algorithms that are more effective than Adaboost at increasing the margin on the training data, but these algorithms have very poor generalization error compared to Adaboost. Also, it is worth noting that when using large decision trees and neural networks as base classifiers, Adaboost works very well even though these base classifiers are known to have high VC-dimensions.

2.3.3 Hierarchical mixture of experts

The HME approach, introduced by Jordan et al in [45], is a tree structured architecture which is based on the principle of divide-and-conquer. The model uses a probabilistic approach to partition the input space into overlapping regions on which the "experts" act. The leaves of the tree are denoted "expert networks", and the non-terminal elements of the tree are called "gating functions". Each gating function is associated with a node (expert network) in the tree. It weights the outputs of the subtrees coming into it, to form a combined output. The final network output is obtained at the root of the tree. An illustration of a HME with 2 levels and a common branching factor of 2 at each level is shown in Figure 1. In the general architecture, multiple levels and branching factors are possible. However, the model structure should be fixed/predetermined before proceeding to training.

Expert network (i, j) produces its output μ_{ij} as a generalized linear function of the input $\mathbf{x}$:

$$\mu_{ij} = f(U_{ij}\mathbf{x}), \tag{59}$$

where U_{ij} is a weight matrix and f is a fixed continuous nonlinearity. For regression problems, f is generally chosen to be the identity function. For binary classification, f is generally taken to be

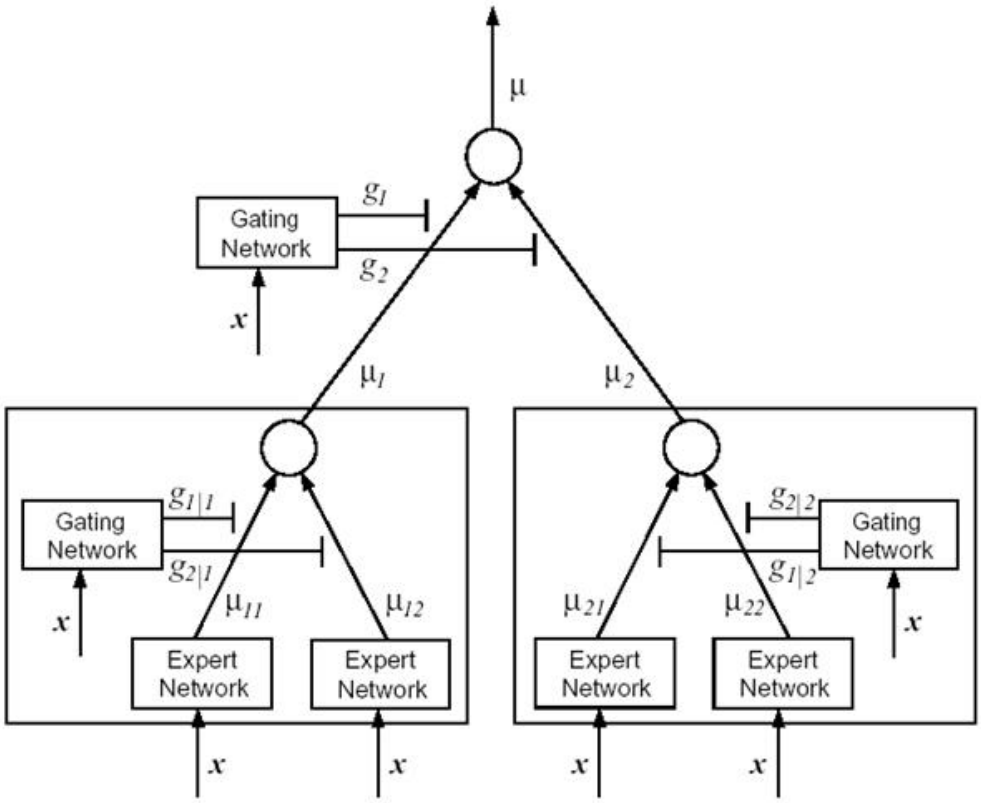


Figure 1. Block diagram of a 2-level HME

the logistic function. Other choices for f could be taken to handle the application at hand (e.g., multiway classification, counting, rate estimation).

The gating networks are also generalized linear functions of the input $\mathbf{x}$. Let $\xi_i = \mathbf{v}_i^T\mathbf{x}$, where $\mathbf{v}_i$ is a weight vector. Then the i^{th} output of the top level gating network is the "softmax" distribution of the $\xi_i's$:

$$g_i = \frac{e^{\xi_i}}{\sum_k e^{\xi_k}}, \tag{60}$$

It is worth noting that the g_i's are positive and sum to one for each $\mathbf{x}$. They can be viewed as a "soft" partitioning of the input space. Similarly, the gating networks at lower levels are generalized linear systems. Thus, we define intermediate variables, ξ_{ij}, as $\xi_{ij} = \mathbf{v}_{ij}^T\mathbf{x}$, and let

$$g_{j|i} = \frac{e^{\xi_{ij}}}{\sum_k e^{\xi_{ik}}} \tag{61}$$

be the output of the j^{th} unit in the i^{th} gating network at the second level of the architecture. The output vector at each nonterminal node of the tree is the weighted sum of the experts below that

node:

$$\mu_{\mathbf{i}} = \sum_j g_{j|i}\mu_{\mathbf{ij}}, \tag{62}$$

and the output at the top level of the tree is:

$$\mu = \sum_i g_i \mu_{\mathbf{i}}. \tag{63}$$

The hierarchy can be given a probabilistic interpretation. First, we refer to g_i and $g_{j|i}$ as the prior probabilities, as they depend only on $\mathbf{x}$. Then, for each expert, we assume that the true output y is chosen from a distribution P with mean μ_{ij}. Therefore, the total probability of generating y from $\mathbf{x}$ is obtained by summing over all the possible paths to the root of the tree, that is:

$$P(y|\mathbf{x}, \theta) = \sum_i g_i \sum_j g_{j|i} P(y|\mathbf{x}, \theta_{ij}). \tag{64}$$

To develop the EM-based learning algorithm for HME, the posterior probabilities associated with each node in the tree are derived according to Bayes' rule:

$$h_i = \frac{g_i \sum_j g_{j|i} P_{ij}(y)}{\sum_i g_i \sum_j g_{j|i} P_{ij}(y)} \tag{65}$$

$$h_{j|i} = \frac{g_{j|i} P_{ij}(y)}{\sum_j g_{j|i} P_{ij}(y)}. \tag{66}$$

Finally, the joint posterior probability h_{ij}, is the product of h_i and $h_{j|i}$:

$$h_{ij} = \frac{g_i g_{j|i} P_{ij}(y)}{\sum_i g_i \sum_j g_{j|i} P_{ij}(y)}. \tag{67}$$

This quantity is the probability that expert network (i, j) has generated the data, based on the knowledge of both the input and the output. It should be emphasized that the above quantities are conditional on the input $\mathbf{x}$.

$$l(\theta, \mathbb{X}) = \sum_k \ln \sum_i g_i^{(k)} \sum_j g_{j|i}^{(k)} P_{ij}(\mathbf{y}^{(k)}). \tag{68}$$

The problem of learning in HME can be casted as a maximum likelihood estimation problem. Given a data set $\mathbb{X} = \{\mathbf{x}_k, y_k\}_{k=1}^N$, its log-likelihood is computed by taking the log of the product of N densities of the form of equation (64). An analytical solution that minimizes the likelihood in (68) could not be derived, as it involves three summations and a logarithm term. In practice, iterative methods such as gradient descent [45], and expectation-maximization [46] algorithms were proposed in the HME literature.

2.3.4 Local fusion

One way to categorize classifier combination methods is based on the way they select or assign weights to the individual classifiers. Some methods are global and assign a degree of worthiness, that is averaged over the entire training data, to each classifier. Other methods are local and adapt the classifiers' performance to different data subspaces.

In [47], a local fusion approach, called Context-Dependent Fusion (CDF), has been proposed. CDF has two main steps. First, it uses a standard clustering algorithm to partition the training signatures into groups of signatures, or contexts. Then, the fusion parameters are adapted to each context. CDF treats the partitioning of the feature space and the selection of local expert classifiers as two independent processes performed sequentially.

In [48], a generic framework for context-dependent fusion, called Context Extraction for Local Fusion (CELF) was proposed. CELF jointly optimizes the partitioning of the feature space and the fusion of the classifiers. The partitioning of the feature space includes a feature discrimination component to identify clusters in subspaces in possibly high-dimensional feature space. Within these subspaces (called also contexts), a linear aggregation of the different algorithms outputs provides a better classification rate than all the individual classifiers and the global approach.

Both CDF and CELF use multiple algorithms and features. Any algorithm/features pair can be used as long as the algorithms have a continuous, probability-like output and the features are metric. However, these approaches are not applicable in the context of sequential data as there is no robust similarity measure between two sequences. Also, the aforementioned approaches do not intervene at the model creation and training level.

2.4 Multiple models fusion

For complex detection and classification problems involving data with large intra-class variations and noisy inputs, good solutions are difficult to achieve, and no single source of information can provide a satisfactory solution. As a result, combination of multiple classifiers (or multiple experts) is playing an increasing role in solving these complex pattern recognition problems, and has proven to be a viable alternative to using a single classifier.

There are many taxonomies for classifier combination methods : trainable vs. non trainable, class label combination vs. continuous output combination, classifier selection vs. classifier fusion, etc.

In the latter taxonomy, classifier selection methods put an emphasis on the development of the classifier structure. First, these methods identify the single best classifier or a selected group of classifiers and then only their outputs are taken as a final decision or for further processing. This approach assumes that the classifiers are complementary, and that their expertise varies according to the different areas of the feature space. For a given test sample, these methods attempt to predict which classifiers are more likely to be correct. Some of these methods consider the output of only a single classifier to make the final decision [49]. Others, combine the output of multiple "local expert" classifiers [50]. Classifier fusion methods operate mainly on the classifiers outputs, and strive to combine the classifiers outputs effectively. This approach assumes that the classifiers are competitive and equally experienced over the entire feature space. For a given test sample, the individual classifiers are applied in parallel, and their outputs are combined in some manner to take a group decision.

Another way to categorize classifier combination methods is based on the way they select or assign weights to the individual classifiers. Some methods are global and assign a degree of worthiness, that is averaged over the entire training data, to each classifier.

2.5 Chapter summary

In this chapter, we presented the background related to the proposed work. We first laid down the fundamentals of Hidden Markov modeling and its use as a sequential data classifier. We presented the different approaches to address complex classification problems with noisy, large, intra-class variable data using fusion methods. We then argued for the need to use multiple models and to efficiently combine their outputs. We also presented several ensemble learning methods that are related to our approach. In the last section of this chapter, we detailed several state-of-the-art methods and approaches for combining multiple models.

CHAPTER 3

ENSEMBLE HMM CLASSIFIER

3.1 Introduction

In this chapter, we start by motivating the need for our eHMM approach. Then, we outline the architecture of the proposed method and its components. The intermediate steps of the eHMM are illustrated using examples from landmine detection application using Ground Penetrating Radar (GPR) signatures and Edge Histogram Descriptors (EHD) features. The details of the application of eHMM to landmine detection using GPR signatures and the results will be described subsequently in detail in chapter 5. The EHD features, along with other features representations for the GPR signatures, are outlined in section 5.3 of the same chapter.

In the following sections, eHMM refers to both the discrete (eDHMM) and continuous (eCHMM) case seemingly, except where otherwise stated (e.g. individual model initialization, clusters' models construction).

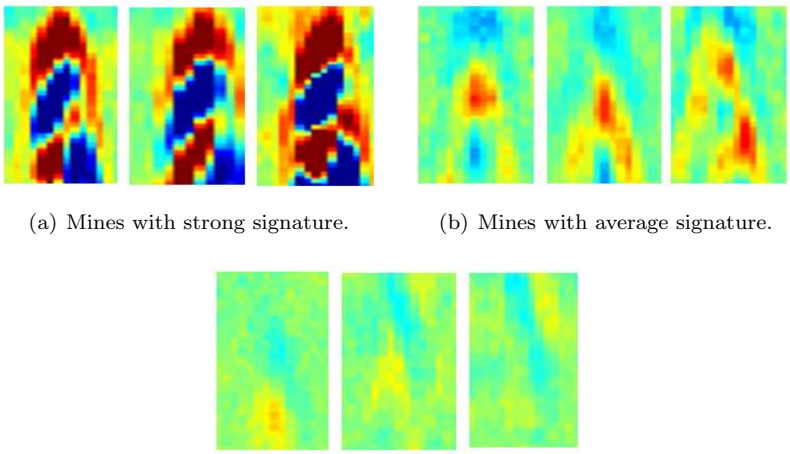
3.2 Motivations

For a two class problem, the baseline HMM models each class by a single model learned from all the observations within that class. The goal is to generalize from all the training data in order to classify unseen observations. However, for complex classification problems, combining observations with different characteristics to learn one model might lead to too much averaging and thus loosing the discriminating characteristics of the observations.

To illustrate this problem, we use an example of detecting buried landmines using GPR sensors[1]. In this case, the training data consists of a set of N GPR alarms labeled as mines (class 1) or clutter (class 0). The goal is to generalize from the training data in order to classify unlabeled GPR signatures. In figure 2, we show three groups of mines with different strengths. It is obvious that grouping all of these signatures, to learn a single model, would lead to poor generalization. Similarly, the false alarms could be caused by different clutter objects and varied environment

[1]The details of the landmine detection application using GPR signatures will be presented in chapter 5

conditions and could have significant variations. The above problem is more acute when data is collected by multiple sensors and/or using various features.



(a) Mines with strong signature. (b) Mines with average signature.

(c) Mines with weak signature.

Figure 2. Mine signatures manually categorized into three groups.

Consequently, learning a set of models that reflect different characteristics of the observations might be more beneficial than using one global model for each class. In this dissertation, we develop a new approach that replaces the two-model classifier with multiple models classifier. For instance, each group of signatures in figure 2 would be used to learn a different model. Our approach aims to capture the characteristics of the observations that would be lost under averaging in the two-model case.

We hypothesize that under realistic conditions, the data are generated by multiple models. The proposed approach, called ensemble HMM (eHMM), attempts to partition the training data in the log-likelihood space and identify multiple models in an unsupervised manner. Depending on the clusters homogeneity and size, a different training scheme is applied to learn the corresponding HMM parameters. In particular, for homogeneous clusters with mainly observations from the same class, the HMM is trained using the standard Baum-Welch algorithm [9]. For clusters containing observations from the two classes, the corresponding HMM is trained using a discriminative training scheme based on the minimization of the misclassification error (MCE) criterion [11]. For clusters with small number of observations, the corresponding HMM parameters are learned using the variational Bayesian approach [29]. The resulting K HMMs are then aggregated through a decision level fusion component to form a descriptive model for the data.

3.3 Ensemble HMM architecture

Let $\mathbb{O}= \{O_r, y_r\}_{r=1}^{R}$ be a set of R labeled sequences of length T where $O_r = \{O_r^{(1)}, \cdots, O_r^{(T)}\}$ and $y_r \in \{1, \cdots, C\}$ is the label (class) of the sequence O_r. The proposed alternative for baseline HMM consists of the construction of a mixture of K HMMs, $\Lambda = \{\lambda_k\}_{k=1}^{K}$, to cover the diversity of the training data. The eHMM has four main components:

1. Similarity matrix computation
2. Pairwise-distance-based clustering
3. Models initialization and training
4. Decision level fusion

These steps are illustrated in the block diagram in figure 3, and are described in the next sections.

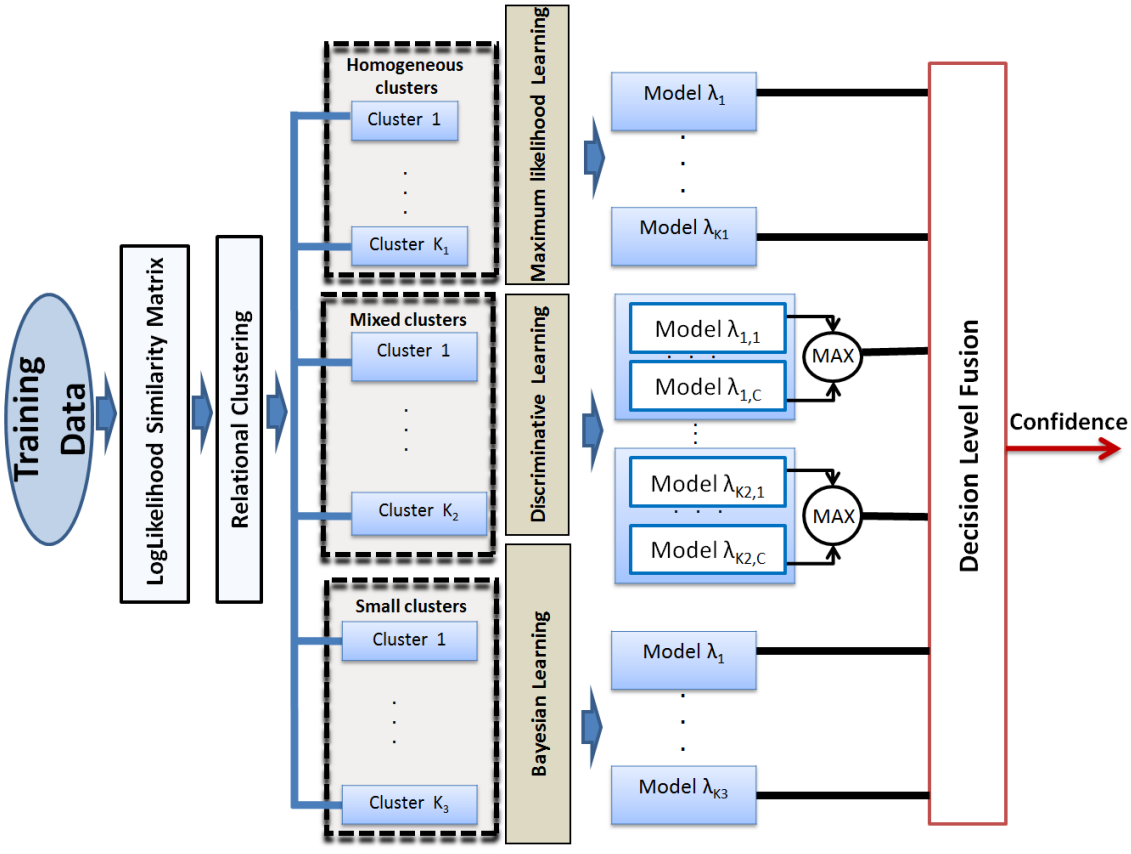


Figure 3. Block diagram of the proposed eHMM (training)

3.3.1 Similarity matrix computation

3.3.1.1 Fitting individual models to sequences

In the first step, each sequence in the training data, O_r, $1 \leq r \leq R$ is used to learn an HMM model λ_r. Even though the use of only one sequence of observations to form an HMM might lead to over-fitting, this step is only an intermediate step that aims to capture the characteristics of each sequence. The formed HMM model is meant to give a maximal description of each sequence and therefore, over-fitting is not an issue in this context. In fact, it is desired that the model perfectly fits the observation sequence. In this case, it is expected that the likelihood of each sequence with respect to its corresponding model is higher than those with respect to the remaining models.

Let $\{\lambda_r^{(0)}\}_{r=1}^R$ be the set of initial models and $s_n^{(r)},\ 1 \leq n \leq N$, be the representative of each state in $\lambda_r^{(0)}$. First, the model states are initialized following one of the methods described in section 2.1.9. In most cases, domain knowledge will be used to initialize the model. This will be illustrated through the example of landmine detection in section 3.3.1.2.

For the discrete case, the codewords $\{v_1, \cdots v_M\}$ of the initial individual DHMM model are the actual observations of the sequence $\{O_1, \cdots O_T\}$. Consequently, the emission probability of each codeword in each state is inversely proportional to their distance to the mean of that state, i.e.,

$$b_n(m) = \frac{\frac{1}{\|v_m - s_n\|}}{\sum_{l=1}^{N} \frac{1}{\|v_m - s_l\|}}, 1 \leq m \leq M. \tag{69}$$

To satisfy the requirement $\sum_{m=1}^{M} b_n(m) = 1$, we normalize the values by:

$$b_n(m) \longleftarrow \frac{b_n(m)}{\sum_{l=1}^{M} b_n(l)}, 1 \leq m \leq M. \tag{70}$$

In the continuous case, the emission probability density functions are modeled by mixtures of Gaussians. In the case of individual sequence models, we use a single component mixture for each state, as the number of observations is small. Thus, the observations belonging to each state are used to estimate the mean and covariance of that state's component.

Then, the Baum-Welch algorithm [9] is used to fit one model to each given observation. Let $\{\lambda_r\}_{r=1}^R$ be the set of trained individual models.

3.3.1.2 Illustrative example: individual model construction step

The different steps of the eHMM will be illustrated using a landmine detection application with GPR data and EHD features. We assume that each signature O_r, as those shown in figure 2, is represented by a sequence of $T = 15$ observations $O_r = \{O_r^{(1)}, O_r^{(2)}, \ldots, O_r^{(15)}\}$. Each observation is

a five-dimensional vector that encodes the strengths of the edges in the horizontal, vertical, diagonal, anti-diagonal, and non-edge directions[2], i.e. $O_r^{(t)} = [H_r^{(t)}, V_r^{(t)}, D_r^{(t)}, A_r^{(t)}, N_r^{(t)}]$. The signatures of the mines shown in figure 2 are characterized by a hyperbolic shape comprised of a succession of rising, horizonal, and falling edges with different strengths of each edge and variable duration in each state. Thus, for each sequence, we build a three-state HMM model that reflects our knowledge of the mine signature. Specifically, the states are labeled state s_1 or diagonal "Dg" state, state s_2 or horizontal "Hz" state, and state s_3 or anti-diagonal "Ad" state.

First, the priors are initialized such that mine models have to start with the diagonal state and non mine models can start in any of the states with equal probabilities. Thus, the priors are initialized as follows:

$$\begin{cases} \Pi = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & \text{if sequence is mine} \\ \Pi = \begin{bmatrix} .33 \\ .33 \\ .33 \end{bmatrix} & \text{if sequence is non-mine} \end{cases} \tag{71}$$

Next, the transition matrix, A, is initialized using prior knowledge (expected latency in each state) and using the left-to-right constraint. We use:

$$A^{(0)} = \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0 & 0.8 & 0.2 \\ 0 & 0 & 1 \end{bmatrix}. \tag{72}$$

To enforce the left-to-right constraint and the prior initialization assumption for the individual mine models, we discard observations with weak diagonal (anti-diagonal) component at the start (end) of the sequence. In other words, we use an adaptive sequence length by defining the new start and end of the sequence, T_{START} and T_{END} using:

$$\begin{cases} T_{START} = min\{t | O_r^{(t)}(D) > \tau, t = 1 \cdots T\} \\ T_{END} = max\{t | O_r^{(t)}(A) > \tau, t = 1 \cdots T\}. \end{cases} \tag{73}$$

In (73), $O_r^{(t)}(D)$ is the diagonal edge component of observation vector $O_r^{(t)}$, $O_r^{(t)}(A)$ is the anti-diagonal edge component of observation vector $O_r^{(t)}$, and τ is a predefined threshold.

[2]The details of the edge features and their extraction will be described in section 5.3

For the emission probabilities initialization, the first step consists of computing the mean of each state. For that matter, we investigate two possible initialization schemes:

- **Initialization 1**: This initialization scheme applies to both the mine and the non mine sequence models. We do not attempt to recognize the state of each observation. Instead, we assume that the first few observations belong to state $s_1^{(r)}$, the middle few observations belong to state $s_2^{(r)}$, and the last few observations belong to state $s_3^{(r)}$. We simply compute the mean of the three states as follows: the mean of the diagonal state "Dg", $s_1^{(r)}$, is the average of observations $(O_{T_{START}} \cdots O_7)$; the mean of the horizontal state "Hz", $s_2^{(r)}$, is the average of observations $(O_6 \cdots O_{10})$; and the mean of the anti-diagonal "Ad" state, $s_3^{(r)}$, is the average of observations $(O_9 \cdots O_{T_{END}})$.

- **Initialization 2**: This initialization scheme differs between mine and non-mine models. For the mine model, we partition its observations into three clusters and label them "Dg", "Ad", and "Hz". The "Dg" cluster corresponds to the partition where the observations have stronger diagonal edges. Similarly, the "Ad" cluster corresponds to the partition where the observations have stronger anti-diagonal edges. The "Hz" cluster corresponds to the partition where the observations have weak diagonal and anti-diagonal edge components. Once each observation is assigned to one of the three clusters, we compute the means, $s_i^{(r)}$, as the average of all observations assigned to cluster $i, 1 \leq i \leq 3$. For the non-mine model, the means of the states are obtained by clustering all of the non target sequence observations into three clusters using the k-means algorithm [24].

In the second step, we compute the initial emission probabilities of each state. For the discrete case, the codewords for the individual DHMM model are the actual observations of the sequence after squeezing $\{O_{T_{START}}, \cdots, O_{T_{END}}\}$. Consequently, the emission probability of each codeword in each state are computed using equations (69) and (70). In the continuous case, the mean and covariance of the emission probability density function for each state's component are estimated using the observations belonging to that state.

3.3.1.3 Computing the similarity matrix

For each observation sequence O_r, $1 \leq r \leq R$, we compute its likelihood in each model λ_p, $Pr(O_r|\lambda_p)$, for $1 \leq p \leq R$, using either the forward or the backward procedure described in section 2.1.6. Since the models share the same architecture, it could be asserted that similar alarms would

have comparable likelihood values. However, the likelihood based similarity may not be always accurate. In fact, some observations can have high likelihood in a visually different observation model. This occurs when most of the elements of a sequence partially match only one or two of the states of the model. In this case, the observation sequence can have a high likelihood in the model but its optimal Viterbi path will deviate from the typical path. To alleviate this problem, we introduce a penalty term, $\mathbf{P}$, to the log-likelihood measure that is related to the mismatch between the most likely sequence of hidden states of the test sequence (O_i) and that of the generating sequence (O_j).

Let $\mathbf{S}$ be the $R \times R$ similarity matrix defined by:

$$\mathbf{S}(i,j) = \alpha \mathbf{L}(i,j) - (1-\alpha)\mathbf{P}(i,j), \tag{74}$$

where

$$\mathbf{L}(i,j) = \log Pr(O_i|\lambda_j) \tag{75}$$

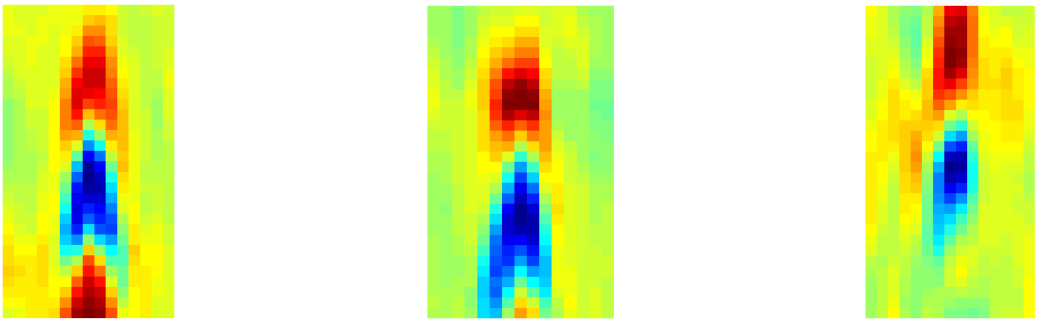
is the log-likelihood of observation sequence i in model λ_j, and

$$\mathbf{P}(i,j) = \sum_{t=1}^{T} \| s^{(j)}_{q_t^{(ji)}} - s^{(j)}_{q_t^{(jj)}} \| \delta(q_t^{(ji)} \neq q_t^{(jj)}), \quad 1 \leq i,j \leq R, \tag{76}$$

is a penalty term that measures the deviation between the Viterbi paths of testing O_i and O_j with model λ_j (recall that model λ_j was learned using observation sequence O_j). In (76), $q_t^{(ji)}$, and $q_t^{(jj)}$ are respectively the most likely (as identified by the Viterbi algorithm [8]) hidden state sequences that generated sequences O_i and O_j from the model λ_j.

The first term in equation (74) is the log-likelihood of sequence O_i being generated from model λ_j. When the likelihood term is high, it is likely that model λ_j generated the sequence O_i. In particular, since by construction model λ_j parameters were adjusted such that they maximize the likelihood of sequence O_j in λ_j, it is expected that the likelihood of each sequence in its corresponding model be high. In this case, sequences O_i and O_j are expected to share the same "dynamics". When the likelihood term is low, it is unlikely that model λ_j generated the sequence O_i. In this case, it is expected that O_i and O_j are not similar.

The second term in equation (74) is a penalty term denoted as the Viterbi path mismatch. It is derived from the Viterbi optimal paths (refer to section 2.1.7). Precisely, an entry P_{ij} is the distance between the optimal path of the sequence O_i in model λ_j and the optimal path of the sequence O_j in λ_j using an appropriate distance measure between paths. For instance, we can use the distance between the mean of the states appearing in each path, as defined in (76). Another distance, commonly used in "string" comparisons, is the "edit distance" [51]. The "edit



(a) Alarm Sig_1 (b) Alarm Sig_2 (c) Alarm Sig_3

Figure 4. GPR signatures of mine alarms

distance" between two strings, say p and q, is defined as the minimum number of single-character edit operations (deletions, insertions, and/or replacements) that would convert p into q. For example, the distance between $p =$ "1111223333" and $q =$ "111223333" is 3. The Viterbi path mismatch term is intended to ensure that similar sequences have few mismatches in their corresponding Viterbi paths. This term is important in applications where the HMM states have a "visual" interpretation or are related to a certain shape characteristic in the original sequence raw data or extracted features. The extra computational cost for obtaining the Viterbi path penalty is very low compared to that of computing the traditional likelihood. Indeed, the Viterbi path is already available when using the forward-backward procedure for the likelihood computation.

The mixing factor $\alpha \in [0, 1]$ of equation (74) is a trade-off parameter between a likelihood-based similarity and a Viterbi-path-mismatch based similarity. It could be either set according to the application at hand or optimized using the available training data and an adequate criterion.

The matrix computed using (74) is a similarity matrix and is not symmetric. Thus, we use the following symmetrization scheme to transform it to a pairwise distance matrix:

$$\begin{cases} 1. \quad \mathbf{D}(i,j) = -\mathbf{S}(i,j) \quad 1 \leq i,j \leq R \\ 2. \quad \mathbf{D}(i,i) = 0, \quad 1 \leq i \leq R \\ 3. \quad \mathbf{D}(i,j) = \max(\mathbf{D}(i,j), \mathbf{D}(j,i)), \quad 1 \leq i,j \leq R. \end{cases} \tag{77}$$

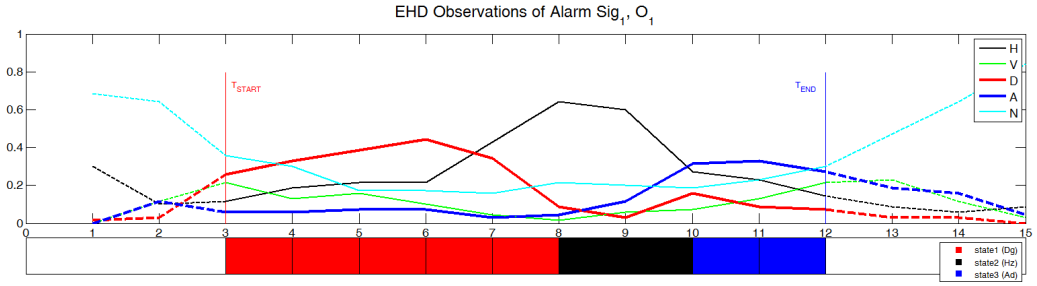
The matrix $\mathbf{D}$ represents the distance or dissimilarity between pairs of sequences O_i and O_j.

3.3.1.4 Illustrative example: similarity matrix computation step

Continuing with the landmine illustrative example, figures 4(a) and 4(b) show two typical mine alarms, Sig_1 and Sig_2, that have similar GPR signatures and similar EHD features. Figure

4(c) shows a mine alarm, Sig_3, that has a visually different signature compared to Sig_1 and Sig_2. Let λ_i be the trained DHMM model for alarm Sig_i and O_i be its observation vector, for $i = 1 \cdots 3$. The initial and trained DHMM models for these 3 alarms are given in figures 5, 6, and 7 respectively.

First, to check the validity of the log-likelihood measure, we compute the probabilities of O_1 and O_2 in their respective models λ_1 and λ_2. As it can be seen in table 1, the probability of each sequence in its own model is higher than its probability in the other sequence's model. Note that since Sig_1 and Sig_2 are similar, the likelihoods L_{12} and L_{21} are relatively high. In fact, both models λ_1 and λ_2 have comparable attributes (codes, states' means, and transition matrices) as shown in figures 5 and 6. On the other hand, since Sig_3 is different from Sig_1 and Sig_2, the likelihoods L_{31} and L_{32} are relatively lower ($L_{31} = -13.23$ and $L_{32} = -11.45$).



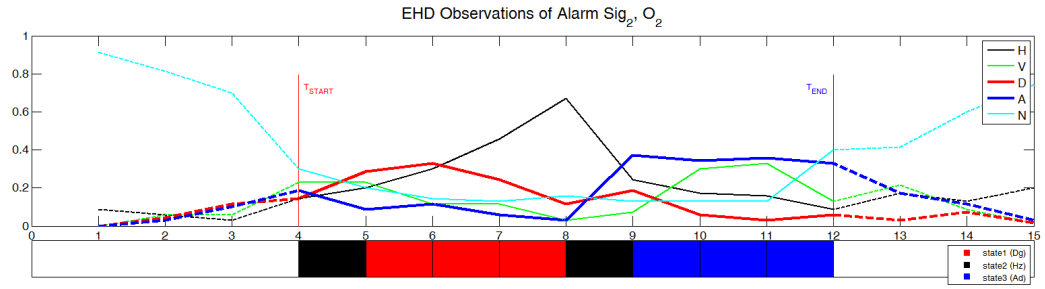
(a)

	States Means					Initial A			A after training		
	H	V	D	A	N	S1	S2	S3	S1	S2	S3
S1	0.23	0.13	0.35	0.06	0.23	0.80	0.20	0.00	0.77	0.23	0.00
S2	0.62	0.04	0.06	0.08	0.21	0.00	0.80	0.20	0.00	0.39	0.61
S3	0.21	0.14	0.10	0.30	0.24	0.00	0.00	1.00	0.00	0.00	1.00
	Observations					Initial B			B after training		
	H	V	D	A	N	S1	S2	S3	S1	S2	S3
	0.11	0.21	0.26	0.06	0.36	0.16	0.06	0.06	0.23	0.00	0.00
	0.19	0.13	0.33	0.06	0.30	0.22	0.02	0.01	0.23	0.00	0.00
	0.21	0.16	0.39	0.07	0.17	0.24	0.00	0.00	0.23	0.00	0.00
	0.21	0.10	0.44	0.07	0.17	0.24	0.00	0.00	0.23	0.00	0.00
	0.43	0.04	0.34	0.03	0.16	0.14	0.19	0.04	0.08	0.39	0.00
	0.64	0.01	0.09	0.04	0.21	0.00	0.73	0.00	0.00	0.61	0.00
	0.60	0.06	0.03	0.11	0.20	0.00	0.00	0.22	0.00	0.00	0.25
	0.27	0.07	0.16	0.31	0.19	0.00	0.00	0.22	0.00	0.00	0.25
	0.23	0.13	0.09	0.33	0.23	0.00	0.00	0.22	0.00	0.00	0.25
	0.14	0.21	0.07	0.27	0.30	0.00	0.00	0.22	0.00	0.00	0.25

(b)

Figure 5. DHMM model λ_1 construction for Sig_1. (a) EHD features of the sequence of 15 observations O_1. T_{START} and T_{END} denote the dynamic range of the sequence and are computed using equation (73). The Viterbi path of testing O_1 with λ_1 is displayed under each observation number. (b) Initial and learned (using Baum-Welch algorithm) parameters of model λ_1.

Table 2 shows the Viterbi paths resulting from the test of each alarm Sig_1, Sig_2, and Sig_3 in each model λ_1, λ_2, and λ_3. We notice that the Viterbi path mismatch penalty is not as determinant as the log-likelihood in this particular illustration test case. Nevertheless, it is worth noting that



(a)

	States Means					Initial A			A after training		
	H	V	D	A	N	S1	S2	S3	S1	S2	S3
S1	0.32	0.15	0.29	0.09	0.16	0.80	0.20	0.00	0.70	0.30	0.00
S2	0.41	0.13	0.13	0.11	0.23	0.00	0.80	0.20	0.00	0.40	0.60
S3	0.16	0.21	0.08	0.35	0.20	0.00	0.00	1.00	0.00	0.00	1.00
	Observations					Initial B			B after training		
	H	V	D	A	N	S1	S2	S3	S1	S2	S3
	0.14	0.23	0.14	0.19	0.30	0.00	0.00	0.19	0.30	0.00	0.00
	0.20	0.23	0.29	0.09	0.20	0.27	0.16	0.02	0.30	0.00	0.00
	0.30	0.11	0.33	0.11	0.14	0.40	0.00	0.00	0.30	0.00	0.00
	0.46	0.11	0.24	0.06	0.13	0.20	0.36	0.01	0.09	0.41	0.00
	0.67	0.03	0.11	0.03	0.16	0.12	0.48	0.02	0.00	0.59	0.01
	0.24	0.07	0.19	0.37	0.13	0.00	0.00	0.19	0.00	0.00	0.25
	0.17	0.30	0.06	0.34	0.13	0.00	0.00	0.19	0.00	0.00	0.25
	0.16	0.33	0.03	0.36	0.13	0.00	0.00	0.19	0.00	0.00	0.25
	0.09	0.13	0.06	0.33	0.40	0.00	0.00	0.19	0.00	0.00	0.25

(b)

Figure 6. DHMM model λ_2 construction for Sig_2. (a) EHD features of the sequence of 15 observations O_2. T_{START} and T_{END} denote the dynamic range of the sequence and are computed using equation (73). The Viterbi path of testing O_2 with λ_2 is displayed under each observation number. (b) Initial and learned (using Baum-Welch algorithm) parameters of model λ_2.

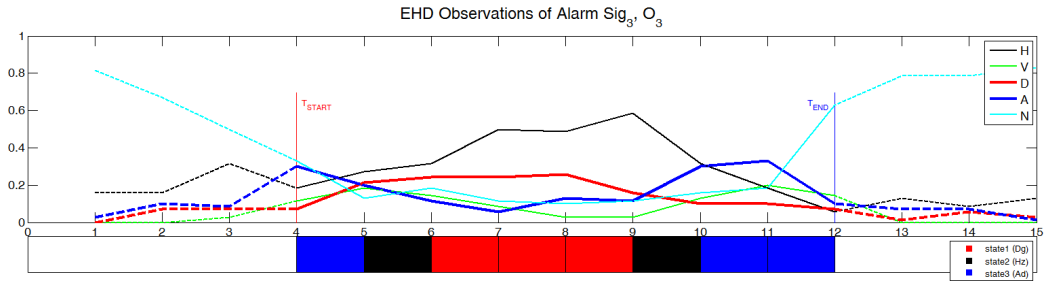
TABLE 1

Log-likelihoods of the three signatures in figure 4 in the models generated by these signatures

L_{ij}	λ_1	λ_2	λ_3
O_1	-1.65	-5.00	-10.13
O_2	-5.50	-1.55	-7.55
O_3	-13.23	-11.45	-1.53

the path p_{11} for Sig_1 in its model shows a perfect symmetry, in terms of diagonal and anti-diagonal states occurrences. However, the path p_{33} for Sig_3 in its model reflects the visual asymmetry of Sig_3 (stronger diagonal component).

In figure 8, we show a non-mine (i.e. clutter) alarm, Sig_4. Let O_4 denote its observation vector and let λ_4 be its DHMM model. The construction of this model is illustrated in figure 9. As mentioned in section 3.3.1.3, the motivation for using the path mismatch penalty comes from the testing of non-mine signatures, such as Sig_4, in a mine model, e.g. λ_1. In fact, in the computation



(a)

	States Means					Initial A			A after training		
	H	V	D	A	N	S1	S2	S3	S1	S2	S3
S1	0.43	0.09	0.25	0.10	0.13	0.80	0.20	0.00	0.80	0.20	0.00
S2	0.30	0.12	0.15	0.14	0.29	0.00	0.80	0.20	0.00	0.00	1.00
S3	0.23	0.15	0.09	0.31	0.22	0.00	0.00	1.00	0.00	0.00	1.00
	Observations					Initial B			B after training		
	H	V	D	A	N	S1	S2	S3	S1	S2	S3
	0.19	0.11	0.07	0.30	0.33	0.00	0.00	0.25	0.20	0.00	0.00
	0.27	0.19	0.21	0.20	0.13	0.08	0.23	0.09	0.20	0.00	0.00
	0.31	0.14	0.24	0.11	0.19	0.13	0.28	0.03	0.20	0.00	0.00
	0.50	0.09	0.24	0.06	0.11	0.26	0.05	0.01	0.20	0.00	0.00
	0.49	0.03	0.26	0.13	0.10	0.29	0.00	0.00	0.20	0.00	0.00
	0.59	0.03	0.16	0.11	0.11	0.19	0.13	0.03	0.00	1.00	0.00
	0.31	0.13	0.10	0.30	0.16	0.00	0.00	0.25	0.00	0.00	0.33
	0.19	0.20	0.10	0.33	0.19	0.00	0.00	0.25	0.00	0.00	0.33
	0.06	0.14	0.07	0.10	0.63	0.05	0.30	0.09	0.00	0.00	0.33

(b)

Figure 7. DHMM model λ_3 construction for Sig_3. (a) EHD features of the sequence of 15 observations O_3. T_{START} and T_{END} denote the dynamic range of the sequence and are computed using equation (73). The Viterbi path of testing O_3 with λ_3 is displayed under each observation number. (b) Initial and learned (using Baum-Welch algorithm) parameters of model λ_3.

TABLE 2

Viterbi paths resulting from testing each alarm in figure 4 by the three models learned from these alarms. p_{ij} refers to the optimal path obtained by testing signature i with model j.

p_{11}	1	1	1	1	2	2	3	3	3	3
p_{12}	1	1	1	2	2	3	3	3	3	
p_{13}	1	1	1	2	2	3	3	3	3	
p_{21}	1	1	1	1	2	2	2	3	3	3
p_{22}	1	1	1	2	2	3	3	3	3	
p_{23}	1	1	1	1	2	2	3	3	3	
p_{31}	1	1	1	1	1	2	2	3	3	3
p_{32}	1	1	1	1	1	2	3	3	3	
p_{33}	1	1	1	1	1	2	3	3	3	

of the log-likelihood L_{41}, most or all of the non-mine alarm observations are assigned to one single code from λ_1 (typically, the one with weak edges), that is the closest code to all the observations of Sig_4. Consequently, most of the observations in O_4 are assigned to the state whose mean is closest to that code. Hence, the probability of O_4 in λ_1 is high ($L_{41} = -1.72$) and its Viterbi path p_{14}, as shown in table 3, is a recurrence of state s_1. In this case, the path mismatch penalties, derived from

the paths shown in table 3, can be easily computed using the "edit distance" ($P_{14} = 9$ and $P_{41} = 6$). Those penalty values are relatively high when compared to the previous penalty values obtained by testing mine alarms in mine models ($P_{12} = 3$, $P_{21} = 4$, $P_{13} = 3$, and $P_{31} = 2$).

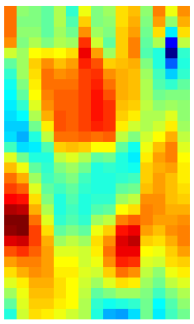
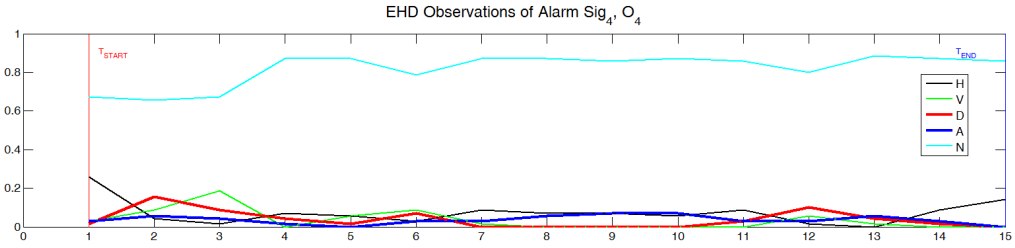


Figure 8. GPR signature for a non-mine alarm, Sig_4



(a)

States Means						Initial A			A after training		
	H	V	D	A	N	S1	S2	S3	S1	S2	S3
S1	0.08	0.06	0.05	0.03	0.78	0.80	0.20	0.00	0.83	0.17	0.00
S2	0.06	0.03	0.01	0.04	0.85	0.00	0.80	0.20	0.00	0.00	1.00
S3	0.07	0.01	0.02	0.04	0.86	0.00	0.00	1.00	0.00	0.00	1.00
Observations						Initial B			B after training		
	H	V	D	A	N	S1	S2	S3	S1	S2	S3
	0.26	0.03	0.01	0.03	0.67	0.09	0.11	0.04	0.17	0.00	0.00
	0.04	0.09	0.16	0.06	0.66	0.20	0.00	0.00	0.17	0.00	0.00
	0.01	0.19	0.09	0.04	0.67	0.20	0.00	0.00	0.17	0.00	0.00
	0.07	0.00	0.04	0.01	0.87	0.02	0.13	0.08	0.17	0.00	0.00
	0.06	0.06	0.01	0.00	0.87	0.03	0.20	0.05	0.17	0.00	0.00
	0.03	0.09	0.07	0.03	0.79	0.20	0.00	0.00	0.17	0.00	0.00
	0.09	0.01	0.00	0.03	0.87	0.01	0.18	0.07	0.00	1.00	0.00
	0.07	0.00	0.00	0.06	0.87	0.00	0.00	0.13	0.00	0.00	0.13
	0.07	0.00	0.00	0.07	0.86	0.00	0.00	0.13	0.00	0.00	0.13
	0.06	0.00	0.00	0.07	0.87	0.00	0.00	0.13	0.00	0.00	0.13
	0.09	0.00	0.03	0.03	0.86	0.01	0.11	0.09	0.00	0.00	0.13
	0.01	0.06	0.10	0.03	0.80	0.20	0.00	0.00	0.00	0.00	0.13
	0.00	0.01	0.04	0.06	0.89	0.00	0.00	0.13	0.00	0.00	0.13
	0.09	0.00	0.01	0.03	0.87	0.01	0.13	0.08	0.00	0.00	0.13
	0.14	0.00	0.00	0.00	0.86	0.04	0.15	0.06	0.00	0.00	0.13

(b)

Figure 9. DHMM model λ_4 construction for Sig_4. (a) EHD features of the sequence of 15 observations O_4. (b) Initial and learned (using Baum-Welch algorithm) parameters of model λ_4.

TABLE 3

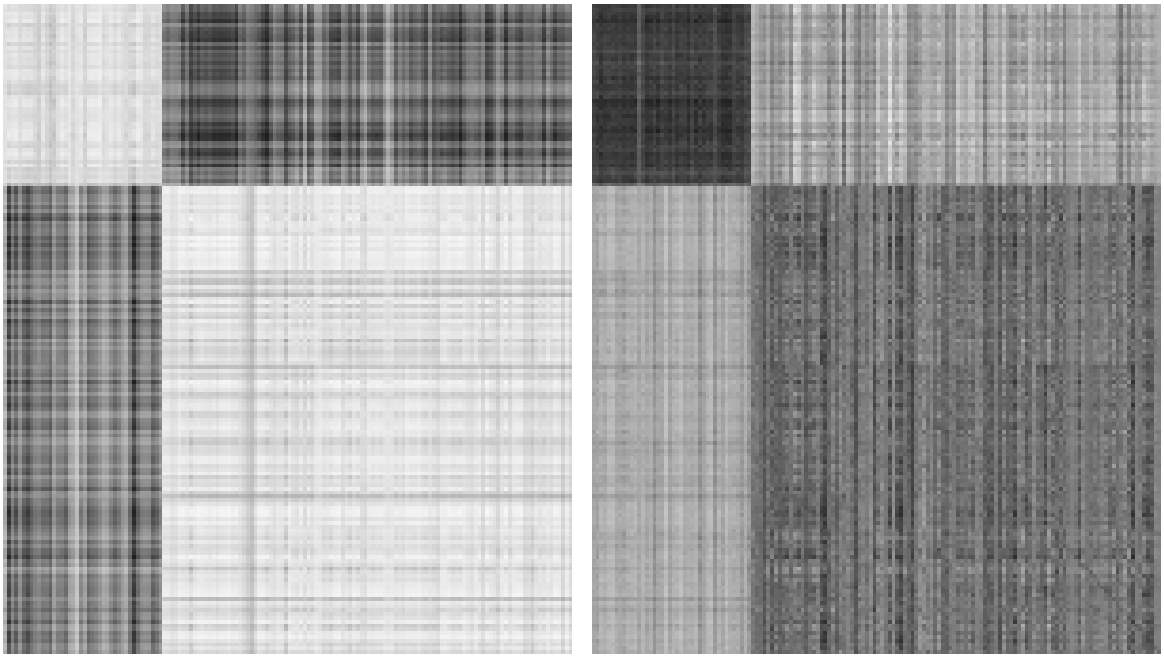
Viterbi paths resulting from testing Sig_1 and Sig_4 in models λ_1 and λ_4. p_{ij} refers to the optimal path obtained by testing signature i with model j.

p_{11}	1	1	1	1	2	2	3	3	3	3					
p_{14}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
p_{44}	1	1	1	1	1	1	2	3	3	3	3	3	3	3	3
p_{41}	1	1	1	1	1	1	1	1	1	1					

Figures 10(a) and 10(b) show the log-likelihood and path mismatch penalty matrices for a large collection of mine and clutter signatures. In these figures, the indices are rearranged so that the first entries correspond to the mine signatures and the latter ones correspond to non-mine signatures. In other words, the diagonal blocks correspond to testing mine signatures in mine models and non-mine signatures in non-mine models, and the off-diagonal blocks correspond to testing mine signatures in non-mine models and non-mine signatures in mine models. In these figures, dark pixels correspond to small values of the log-likelihood and path mismatch penalty and bright pixels correspond to larger entries of the corresponding matrices. Note that in the case of the log-likelihood matrix of figure 10(a), the diagonal blocks are brighter than the off-diagonal blocks. Conversely, in the path mismatch penalty matrix of figure 10(b)), the diagonal blocks are darker than the off-diagonal blocks. Thus, both the log-likelihood and the path mismatch penalty provide complementary measures for the (dis)similarity between signatures. This was the motivation for combining both matrices using the mixing factor α in (74).

3.3.2 Pairwise-distance-based clustering

The similarity matrix, computed using equation (74) and illustrated in figure 10 for the case of landmine detection, reflects the degree to which pairwise signatures are considered similar. The largest variation is expected to be between classes. Other significant variation may exist within the same class, e.g. the groups of signatures shown in figure 2. Our goal is to identify the similar groups so that one model can be learned for each group. This task can be achieved using any relational clustering algorithm. In our work, we investigate three different clustering algorithms : hierarchical [24], spectral with learnable cluster dependent kernels [52], and fuzzy clustering with multiple kernels [53].



(a) Log-likelihood (b) path mismatch penalty

Figure 10. Log-likelihood and path mismatch penalty matrices for a large collection of mine and clutter signatures

3.3.2.1 Hierarchical Clustering

In this case, the natural choice is to use an agglomerative hierarchical algorithm [24]. Agglomerative hierarchical clustering is a bottom-up approach that starts with each data point as a cluster. Then, it proceeds by merging the most similar clusters to produce a sequence of clusters. Several similarity measures between clusters have been used [24]. Examples include single link, complete link, average link, and ward distance. In the single link method, the distance between two clusters is the minimum of the distances between all pairs of sequences drawn from the two clusters. In the complete link based algorithm, the distance between two clusters is the maximum of all pairwise distances between sequences in the two clusters. Complete link method tends to produce compact clusters, while single link is known to result in "elongated" clusters. Another interesting similarity measure between two clusters is the minimum-variance distance, or ward distance [54]. This distance is defined as

$$d(i,j) = \frac{n_i n_j}{n_i + n_j} \|\mathbf{c_i} - \mathbf{c_j}\|^2 \tag{78}$$

where n_k and $\mathbf{c_k}$ are respectively the cardinality and the centroid of the cluster C_k. It has been shown in [55] that this approach merges the two clusters that lead to the smallest increase in the overall variance.

3.3.2.2 Spectral Clustering with Learnable Cluster Dependent Kernels (FLeCK)

FLeCK [52] is a kernel clustering algorithm that works on both object based representation and relational data. It learns multiple kernels by simultaneously optimizing intra-cluster and inter-cluster distances. Specifically, it tries to simultaneously minimize:

$$J^{intra} = \sum_{i=1}^{K}\sum_{j=1}^{N}\sum_{k=1}^{N}\beta_{jk}^{i}(1 - exp(-\frac{\mathbf{r}_{jk}}{\sigma_i})) - \sum_{i=1}^{K}\frac{\rho}{\sigma_i} \tag{79}$$

and maximize

$$J^{inter} = \sum_{i=1}^{K}\sum_{j=1}^{N}\sum_{k=1}^{N}\alpha_{jk}^{i}exp(\frac{\mathbf{r}_{jk}}{\sigma_i}) - \sum_{i=1}^{K}\frac{\rho}{\sigma_i}. \tag{80}$$

In (79) and (80), $\mathbf{r}$ is a distance matrix representing the degree to which objects from a dataset $\{x_1, ...x_N\}$ are dissimilar; σ_i is the scaling parameter of cluster i; K is the number of clusters; ρ is a regularization constant; β_{jk}^{i} is the likelihood that two points x_j and x_k belong to the same cluster; and α_{jk}^{i} is the likelihood that two points x_j and x_k belong to different clusters. Precisely,

$$\begin{cases} \beta_{jk}^{i} = u_{ij}^{m}u_{ik}^{m} \\ \alpha_{jk}^{i} = u_{ij}^{m}(1 - u_{ik}^{m}) + u_{ik}^{m}(1 - u_{ij}^{m}) \end{cases} \tag{81}$$

where $m \in (1, \infty)$ is a constant that determines the level of fuzziness, and u_{ij} is the fuzzy membership of $\mathbf{x}_j$ in cluster i and satisfies

$$0 \leq u_{ij} \leq 1 \quad and \quad \sum_{i=1}^{K} u_{ij} = 1, \quad for \quad j = 1, \cdots, N. \tag{82}$$

The first term in (79) seeks clusters that have compact local relational distances

$$\mathbf{D}_{jk}^{i} = 1 - exp(-\frac{\mathbf{r}_{jk}}{\sigma_i}) \tag{83}$$

with respect to each cluster i.

Assuming that the scaling parameters σ_i's are independent from each other, it can be shown [52] that optimization of (79) and (80) can be performed iteratively using

$$\sigma_i^{(t)} = \frac{\sum_{j=1}^{N}\sum_{k=1}^{N}\beta_{jk}^{i}\mathbf{r}_{jk}^{2}exp(-\frac{\mathbf{r}_{jk}}{\sigma_i^{(t-1)}})}{\sum_{j=1}^{N}\sum_{k=1}^{N}\alpha_{jk}^{i}\mathbf{r}_{jk}exp(-\frac{\mathbf{r}_{jk}}{\sigma_i^{(t-1)}}) + \sum_{j=1}^{N}\sum_{k=1}^{N}\beta_{jk}^{i}\mathbf{r}_{jk}exp(-\frac{\mathbf{r}_{jk}}{\sigma_i^{(t-1)}})} \tag{84}$$

and

$$u_{ij} = \frac{1}{\sum_{t=1}^{K}(\frac{d_{ij}^{2}}{d_{tj}^{2}})^{\frac{1}{m-1}}}, \tag{85}$$

where d_{ik}^{2} is the Euclidean distance from feature $\mathbf{x}_k$ to cluster i that can be written [56] in terms of the relational matrix $\mathbf{D}^{i}$ as

$$d_{ik}^{2} = (\mathbf{D}^{i}\mathbf{v}_i)_k - \frac{\mathbf{v}_i^{t}\mathbf{D}^{i}\mathbf{v}_i}{2}. \tag{86}$$

In (86), $\mathbf{v}_i$ is the membership vector of all N samples in cluster i defined by

$$\mathbf{v}_i = \frac{(u_{i1}^m, \cdots, u_{iN}^m)^t}{\sum_{j=1}^N u_{ij}^m}. \tag{87}$$

Starting with an initial partition, FLeCK iteratively updates the fuzzy membership variables β_{jk}^i and α_{jk}^i and the scaling parameters σ_i. The FLeCK algorithm steps are summarized in Algorithm 4.

Algorithm 4 The FLeCK algorithm, adapted from [52]

Require:
Symmetric distance, **r**
Number of clusters K, fuzzifier m, and scaling parameters σ_i
Ensure:
repeat
    Compute the distance D^i for all clusters using (83);
    Compute the membership vectors $\mathbf{v}_i$ using (87);
    Compute the distances using (86);
    Update the fuzzy memberships using (85);
    Update the scaling parameter σ_i using (84);
until Fuzzy memberships do not change

3.3.2.3 Fuzzy Clustering with multiple kernels (FCM-MK)

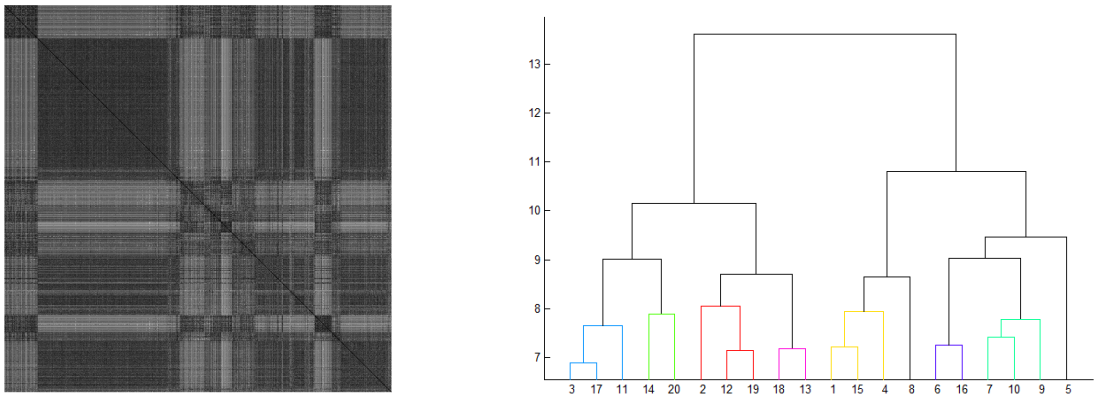
The FCM-MK [53] consists of reducing the dimensionality of the original distance matrix and performing fuzzy c-means clustering [57] on the reduced data. The steps for FCM-MK are detailed in Algorithm 5.

Algorithm 5 The FCM-MK algorithm, adapted from [53]

Require: symmetric distance matrix, D
Require: "neighborhood" size
Ensure:
Compute σ_K matrix using the "neighborhood" parameter
Compute Kernel Matrix K as $K = exp(-((D^2)/\sigma_K))$
Normalize K
Perform singular value decomposition (svd) on normalized kernel K
Derive the corresponding mapping using the corresponding feature reduction dimension
Perform fuzzy c-means clustering on the reduced dimension space mapping
Derive the corresponding partition

3.3.2.4 Illustrative example: hierarchical clustering step

Continuing with the illustrative example of landmine detection, we use the standard hierarchical clustering algorithm to cluster the distance matrix **D** computed from the similarity matrix **S** of (74), using (77). The log-likelihood and path mismatch penalty terms of **S** for a collection of



(a) Distance matrix (b) Dendrogram

Figure 11. Hierarchical clustering results of the matrix **D** for a large collection of mine and clutter signatures

mine and clutter signatures were shown in figure 10. We cluster **D** into a pre-set number of clusters, $K = 20$. The choice of K stems from prior knowledge of the training data at hand. In particular, the choice of K should depend on the number of mine types, the clutter categories, and burial depths observed in the data collection. In Figure 11(a), we plot the resulting distance matrix (that combines figure 10(a) and figure 10(b)) as a gray-scale intensity image. We use the VAT algorithm [58] to rearrange the matrix entries so that each diagonal block corresponds to a cluster. The diagonal blocks of the matrix are darker, which corresponds to smaller intra-cluster distances.

Similarly, the dendrogram in figure 11(b) shows that, at a certain threshold, we can identify two main groups of clusters. On the left hand side of the dendrogram, clusters $\{3, 17, \cdots, 13\}$ have mainly clutter signatures. On the right hand side, clusters $\{1, 5, \cdots, 15\}$ are dominated by mine signatures. These findings are further highlighted in figure 12 where some clusters (e.g. cluster 1, 5, and 15) have a large number of mines and few or no clutter alarms. These are typically strong clutter alarms with mine-like signatures. Figure 13 shows sample alarms belonging to cluster 5.

Similarly, some clusters are dominated by clutter with few mines (e.g. cluster 2, 13, and 18). The few mines included in these clusters, as shown in figure 14, are typically mines with weak signatures that are either low metal mines or mines buried at deep depths. Other clusters are composed of a mixture of mines and clutter signatures (e.g. cluster 3, 6, and 11). The mines within these clusters are either low metal mines (Figure 12 (b), cluster 6) or mines buried at deep depths (Figure 12 (c), clusters 3 and 11). Sample alarms belonging to cluster 11 are shown in figure 15.

The above figures illustrate the purpose of step 2 of the proposed eDHMM approach, i.e.

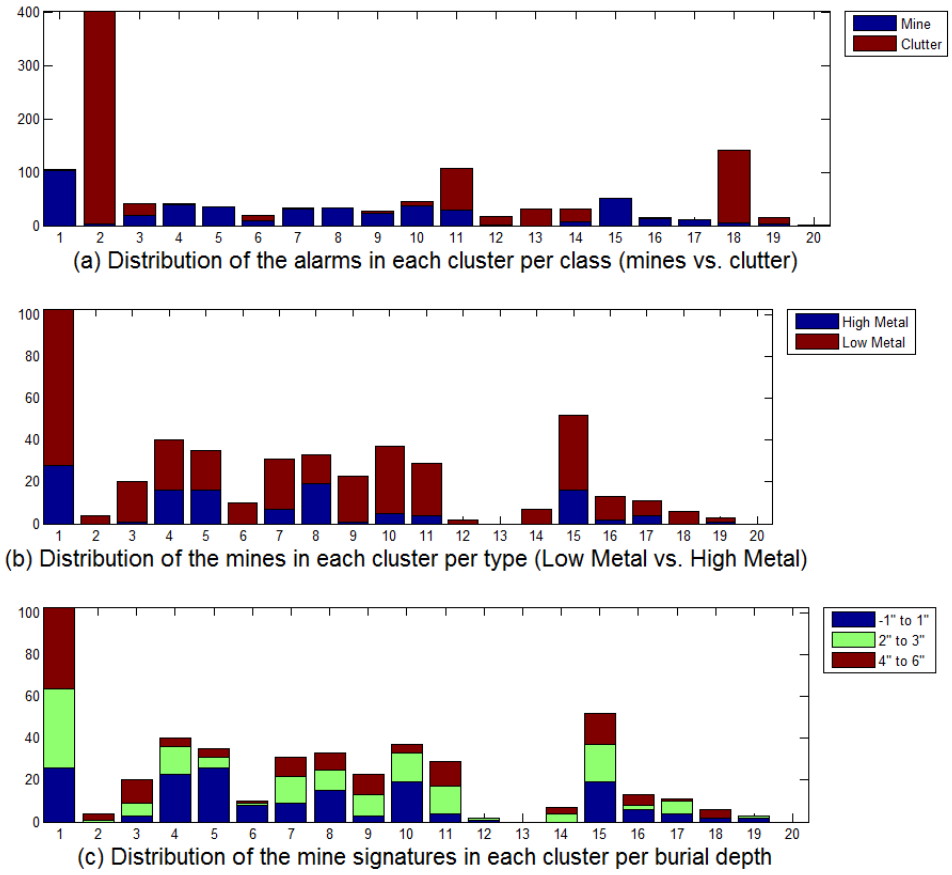


Figure 12. Distribution of the alarms in each cluster: (a) per class, (b) per type, (c) per depth.

grouping similar signatures into clusters.

To summarize, using the similarity matrix $\mathbf{D}$ of (77), the R sequences are clustered into K clusters. The objective of this clustering step is to group similar observations in the log-likelihood space. The observations forming each cluster are then used to learn multiple HMM models. The initialization of each model is performed by averaging the parameters of the individual models of the sequences belonging to that cluster. Then, for each cluster, the initial model and the sequences belonging to that cluster are used to train and update the HMM model parameters. In the next step, we show how different HMM models can be learned for the different clusters.

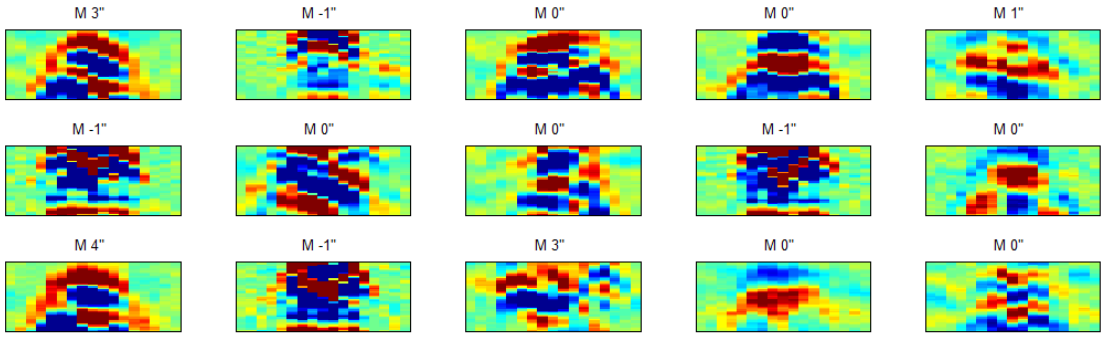


Figure 13. Sample signatures belonging to a cluster dominated by strong mines (cluster 5). Above each signature, "M" denotes a mine signature and the number refers to the burial depth.

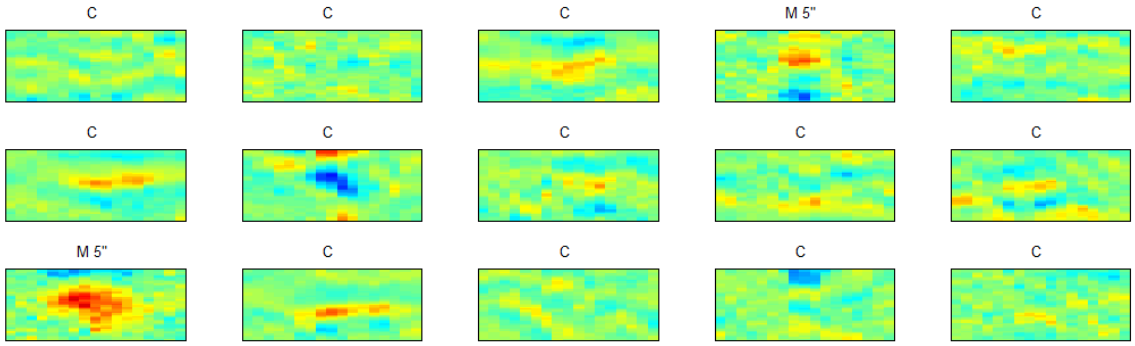


Figure 14. Sample signatures belonging to a cluster dominated by weak signatures (cluster 18). Above each signature, "M" denotes a mine signature, "C" denotes a clutter signature, and the number refers to the burial depth.

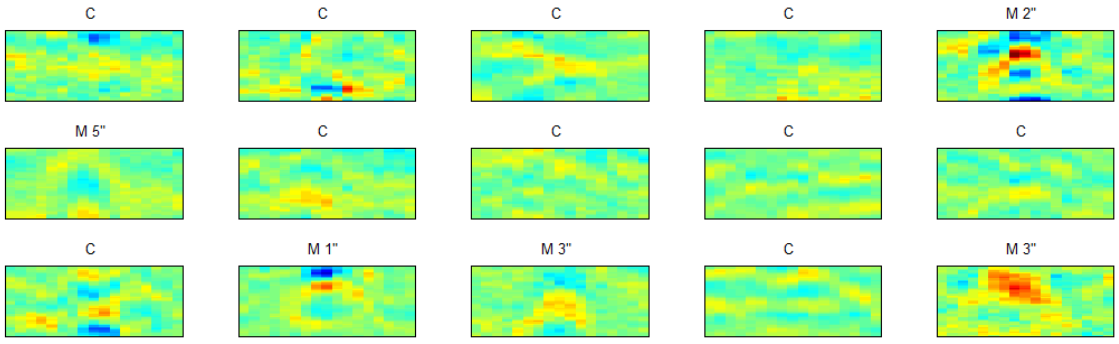


Figure 15. Sample signatures belonging to a cluster that has a mixture of weak mines and clutter signatures (cluster 11). Above each signature, "M" denotes a mine signature, "C" denotes a clutter signature, and the number refers to the burial depth.

3.3.3 Initialization and training of the ensemble HMM

The previous clustering step results in K clusters, each comprised of N_k potentially similar sequences. These K clusters will be used to learn the ensemble HMMs. Let $N_k^{(c)}$ denote the number of sequences assigned to the same cluster k and labeled as class c, such that

$$0 \leq N_k^{(c)} \leq N_k, \quad and \quad \sum_{c=1}^{C} N_k^{(c)} = N_k. \tag{88}$$

For instance, for the landmine example, if we let $c = 1$ denote the class of mines and $c = 0$ denote the class of clutter, $N_k^{(1)}$ would be the number of mines assigned to cluster k. The goal of this step is to initialize up to C HMM models $\{\lambda_k^{(c)}\}$ for each of the K clusters. Let $\mathbb{O}_k^{(c)} = \{O_r^{(c)}, y_r^{(c)}\}_{r=1}^{N_k^{(c)}}$ be the set of sequences of class c belonging to cluster k and $\{\lambda_r^{(c)}\}_{r=1}^{N_k^{(c)}}$ be the corresponding individual sequence models, $c \in \{1, \cdots, C\}$.

For each cluster, we devise the following optimized training methods based on the cluster's size and homogeneity.

- **Clusters dominated by sequences from only one class:** In this case, we learn only one model for this cluster. The sequences within this cluster are presumably similar and belong to the same ground truth class, denoted C_i. We assume that this cluster is a representative of that particular dominating class. It is expected that the class conditional posterior probability is uni-modal and peaked around the MLE of the parameters. Thus, a maximum likelihood estimation would result in an HMM model parameters that best fit this particular class. For these reasons, we use the standard Baum-Welch re-estimation procedure [9]. Let K_1 be the number of homogenous clusters and $\{\lambda_i^{(C_i)}, i = 1, \cdots K_1\}$ denote the set of BW-trained models.

- **Clusters with a mixture of observations belonging to different classes:** In this case, we learn one model for each class represented in this cluster. It is expected that the posterior distribution of the classes is multimodal. The MLE approach is not adequate and more advanced techniques are needed to address the multimodality, such as genetic algorithms [59], or simulated annealing optimization [60]. In our work, we build a model for each class within the cluster. The focus is on finding the class boundaries within the posteriors rather than trying to approximate a joint posterior probability. Thus, the models parameters are jointly optimized such that the overall misclassification error is minimized. In this context, we use discriminative training based on minimizing the misclassification error to learn a model for each class [11]. The details of the discriminative training based on the MCE criterion are given in section

2.1.8.2. Let K_2 be the number of mixed clusters and $\{\lambda_j^{(c)}, j = 1, \cdots K_2, c = 1, \cdots C\}$ be the set of MCE-trained models.

- **Clusters containing a small number of sequences:** In this case, we learn only one model for the dominating class, denoted C_k, of each cluster. The MLE and MCE approaches need a large number of data points to give good estimates of the model parameters. Thus, when a cluster has few samples, the above approaches may not be reliable. The Bayesian training framework [25] , on the other hand is suitable for clusters with small number of sequences. These clusters may contain information about sequences with distinctive characteristics. In this case, we use a variational Bayesian approach [29] to approximate the class conditional posterior distribution. First, the models parameters are given priors. Then, parameter posteriors are computed given the available data. Finally, the required probabilities are derived by integrating over the parameters posteriors. Let K_3 be the number of small clusters and $\{\lambda_k^{(C_k)}, k = 1, \cdots K_3\}$ denote the set of Bayesian-trained models.

Therefore, for each homogenous cluster i, we define one model $\lambda_i^{(C_i)}$, $i = 1, \cdots, K_1$, for the dominating class C_i. For mixed clusters, we define C models per cluster: $\lambda_j^{(c)}$, $c = 1 \ldots C$, $j = 1, \cdots, K_2$. For each small cluster, we define one model $\lambda_k^{(C_k)}$ for the dominating class C_k. The ensemble HMM mixture is defined as $\{\lambda_k^{(c)}\}$, where $k \in \{1, \cdots, K\}$, and $c = C_k$ if cluster k is dominated by sequences labeled with class C_k, and $c \in \{1 \cdots, C\}$ if cluster k is a mixed cluster.

We assume that all models $\lambda_k^{(c)}$ have a fixed number of states N. For each model $\lambda_k^{(c)}$, the initialization step consists of assigning the priors, the initial states transition probabilities, and the states parameters (initial means and initial emission probabilities) using the observations $O_r^{(c)}$ and their respective individual models $\lambda_r^{(c)}$, $r \in \{1, \cdots, N_k^{(c)}\}$. In particular, the initial values for the priors and the state transition probabilities are obtained by averaging, respectively, the priors and the state transition probabilities of the individual models $\lambda_r^{(c)}, r \in \{1, \cdots, N_k^{(c)}\}$. The initialization of the emission probabilities in each state, $b_n^{(k,c)}$, depends on whether the HMM is discrete or continuous.

- Discrete HMM (DHMM): the state representatives and the codebook of model $\lambda_k^{(c)}$ are obtained by partitioning and quantizing the observations $\mathbb{O}_k^{(c)}$. Any clustering algorithm, such as the k-means [61] or the fuzzy c-means [62], could be used for this task. First, the class-(c) sequences belonging to cluster k, $O_r^{(c)}$, are "unrolled" to form a vector of observations $\mathbf{U}^{(k,c)}$ of length $N_k^{(c)}T$. The state representatives, $s_n^{(k,c)}$'s, are obtained by clustering $\mathbf{U}^{(k,c)}$ into N

clusters and taking the centroid of each cluster as the state representative. Similarly, the codebook $\mathbf{V}^{(k,c)} = [v_1^{(k,c)}, \cdots, v_M^{(k,c)}]$ is obtained by clustering $\mathbf{U}^{(k,c)}$ into M clusters. For each symbol $v_m^{(k,c)}$, the membership in each state $s_n^{(k,c)}$ is computed using:

$$b_n(m) = \frac{\frac{1}{\|v_m^{(k,c)} - s_n^{(k,c)}\|}}{\sum_{l=1}^{N} \frac{1}{\|v_m^{(k,c)} - s_l^{(k,c)}\|}}, 1 \leq m \leq M. \tag{89}$$

To satisfy the requirement $\sum_{m=1}^{M} b_n(m) = 1$, we scale the values by:

$$b_n^{(k,c)}(m) \longleftarrow \frac{b_n^{(k,c)}(m)}{\sum_{l=1}^{M} b_n^{(k,c)}(l)} \tag{90}$$

- Continuous HMM (CHMM): we assume that each state has N_g Gaussian components. For each model $\lambda_k^{(c)}$, the observation sequences $O_r^{(c)}$, $r \in \{1, \cdots, N_k^{(c)}\}$, are assigned to one of the states $s_n^{(k,c)}$ and unrolled to form a vector of observations $\mathbf{U}_n^{(k,c)}$, for each state $s_n^{(k,c)}$, $n = 1, \cdots, N$. The mean and covariance of component g of state $s_n^{(k,c)}$ are computed by clustering $\mathbf{U}_n^{(k,c)}$ into N_g clusters using the k-means algorithm [23]. Precisely, the mean of each component is the center of one the resulting clusters and the covariance is estimated using the observations belonging to that same cluster.

After initialization, we use one of the training schemes described earlier, to update $\lambda_k^{(c)}$ parameters using the respective observations $\mathbb{O}_k^{(c)}$, $k \in \{1, \cdots, K\}$, $c \in \{1, \cdots, C\}$. To summarize, for homogenous clusters, BW training results in one model λ^{BW} per cluster. For mixed clusters, MCE training results in C models per cluster: $\lambda_c^{MCE}, c = 1 \ldots C$. For small clusters, variational Bayesian learning results in one model per cluster, λ^{VB}.

3.3.3.1 Illustrative example: cluster models initialization and training steps

Referring back to the landmine example, the clustering of the training data resulted in 20 clusters. As it was shown in figure 12, eight clusters (1, 4, 5, 7, 8, 15, 16, and 17) were composed of only mine alarms, two clusters (3 and 20) contain exclusively clutter alarms, and the remaining ten clusters have both mine and clutter alarms. Therefore, using the notations of Algorithm 6, we define our eHMM as

$$\begin{cases} \lambda_i^{BW} = \{\lambda_1^{(M)}, \lambda_4^{(M)}, \lambda_5^{(M)}, \lambda_7^{(M)}, \lambda_8^{(M)}, \lambda_{15}^{(M)}, \lambda_{16}^{(M)}, \lambda_{17}^{(M)}, \lambda_{13}^{(C)}, \lambda_{20}^{(C)}\}, \\ \lambda_j^{MCE} = \{\lambda_j^{(M)}, \lambda_j^{(C)}\}, j \in \{2, 3, 6, 9, 10, 11, 12, 14, 16, 18, 19\}, \\ \lambda_k^{VB} = \emptyset. \end{cases} \tag{91}$$

Depending on the cluster type, one or two HMM models are built based on the elements belonging to that cluster: one for mines and/or one for clutter. The initialization of these models is performed as follows. The priors and transition matrices of model $\{\lambda_k^{(M)}\}$ and $\{\lambda_k^{(C)}\}$ are obtained by averaging the corresponding parameters of the individual models of the mine and clutter signatures belonging to cluster k, respectively. We assume that each model has $N = 3$ states. The initialization of the emission probabilities for the discrete case is straightforward. That is, the means of the states s_n, $n = 1, \cdots, N$, are obtained by clustering the codes of the individual models into $N = 3$ clusters using the k-means algorithm [23]. Similarly, the codes v_m, $m = 1, \cdots, M$, of the cluster model are obtained by clustering the codes of the individual models into $M = 20$ clusters.

TABLE 4

λ_5^M DHMM model parameters of cluster 5

Means						**Initial A**			**A**		
	H	V	D	A	N	s_1	s_2	s_3	s_1	s_2	s_3
s_1	0.28	0.15	0.31	0.10	0.16	0.67	0.33	0.00	0.59	0.41	0.00
s_2	0.48	0.10	0.12	0.13	0.17	0.00	0.69	0.31	0.00	0.26	0.74
s_3	0.27	0.15	0.09	0.34	0.16	0.00	0.00	1.00	0.00	0.00	1.00
Codes						**Initial B**			**B**		
v_1	0.21	0.23	0.09	0.34	0.14	0.00	0.00	0.22	0.00	0.00	0.12
v_2	0.36	0.01	0.10	0.04	0.39	0.03	0.00	0.01	0.04	0.09	0.03
v_3	0.26	0.00	0.06	0.00	0.67	0.01	0.00	0.01	0.05	0.07	0.03
v_4	0.57	0.01	0.06	0.07	0.23	0.00	0.11	0.02	0.02	0.15	0.01
v_5	0.09	0.53	0.04	0.26	0.13	0.00	0.00	0.02	0.00	0.00	0.12
v_6	0.47	0.09	0.21	0.10	0.11	0.03	0.19	0.00	0.03	0.14	0.01
v_7	0.36	0.17	0.33	0.11	0.07	0.18	0.03	0.00	0.15	0.00	0.00
v_8	0.38	0.09	0.11	0.26	0.09	0.01	0.16	0.26	0.00	0.00	0.12
v_9	0.00	0.56	0.01	0.11	0.31	0.00	0.00	0.00	0.00	0.00	0.12
v_{10}	0.69	0.03	0.09	0.06	0.09	0.00	0.20	0.00	0.03	0.13	0.02
v_{11}	0.27	0.17	0.27	0.13	0.16	0.17	0.01	0.00	0.14	0.01	0.00
v_{12}	0.57	0.06	0.19	0.11	0.04	0.00	0.28	0.00	0.03	0.14	0.02
v_{13}	0.20	0.13	0.24	0.10	0.31	0.11	0.00	0.01	0.09	0.04	0.02
v_{14}	0.09	0.23	0.23	0.07	0.37	0.08	0.00	0.00	0.07	0.04	0.03
v_{15}	0.14	0.14	0.11	0.10	0.50	0.03	0.00	0.08	0.06	0.06	0.04
v_{16}	0.14	0.11	0.06	0.47	0.11	0.00	0.00	0.19	0.00	0.00	0.12
v_{17}	0.19	0.14	0.41	0.07	0.13	0.19	0.00	0.00	0.15	0.00	0.00
v_{18}	0.10	0.29	0.04	0.33	0.33	0.00	0.00	0.16	0.00	0.00	0.12
v_{19}	0.14	0.26	0.30	0.05	0.23	0.12	0.00	0.00	0.10	0.03	0.02
v_{20}	0.39	0.09	0.21	0.04	0.26	0.05	0.01	0.00	0.06	0.09	0.01

After initialization, and depending on the cluster size and homogeneity, one of the training schemes described earlier, is devised. In table 4, we show the initial and the BW-trained model of cluster 5. Recall that cluster 5 contains a large number of strong mines and few clutter alarms with mine-like signatures, as it was shown in figure 13. Therefore, model $\lambda_5^{(M)}$ is expected to reflect the

TABLE 5

λ_2^M DHMM model parameters of cluster 2

Means						**Initial A**			**A**		
	H	V	D	A	N	s_1	s_2	s_3	s_1	s_2	s_3
s_1	0.18	0.11	0.22	0.09	0.39	0.78	0.22	0.00	0.82	0.18	0.00
s_2	0.14	0.17	0.08	0.10	0.51	0.00	0.00	1.00	0.00	0.00	1.00
s_3	0.17	0.15	0.03	0.27	0.38	0.00	0.00	1.00	0.00	0.00	1.00
Codes						**Initial B**			**B**		
v_1	0.21	0.10	0.27	0.07	0.34	0.11	0.00	0.00	0.20	0.00	0.00
v_2	0.01	0.09	0.09	0.11	0.70	0.00	0.00	0.07	0.00	0.00	0.08
v_3	0.03	0.27	0.00	0.10	0.60	0.00	0.00	0.07	0.00	0.00	0.08
v_4	0.06	0.44	0.04	0.14	0.31	0.11	0.00	0.00	0.00	0.00	0.08
v_5	0.09	0.39	0.03	0.10	0.40	0.11	0.00	0.00	0.00	0.00	0.08
v_6	0.04	0.07	0.06	0.19	0.64	0.00	0.00	0.07	0.00	0.00	0.08
v_7	0.26	0.17	0.06	0.16	0.36	0.00	0.00	0.07	0.00	0.00	0.08
v_8	0.09	0.31	0.02	0.28	0.30	0.00	0.00	0.13	0.00	0.00	0.08
v_9	0.26	0.09	0.04	0.21	0.40	0.00	0.00	0.07	0.00	0.00	0.08
v_{10}	0.04	0.13	0.14	0.06	0.63	0.11	0.00	0.00	0.05	0.21	0.01
v_{11}	0.24	0.01	0.00	0.03	0.71	0.11	0.00	0.07	0.05	0.17	0.02
v_{12}	0.21	0.09	0.40	0.01	0.29	0.11	0.00	0.00	0.20	0.00	0.00
v_{13}	0.30	0.01	0.36	0.03	0.30	0.11	0.00	0.00	0.20	0.00	0.00
v_{14}	0.50	0.03	0.11	0.06	0.30	0.00	0.50	0.00	0.09	0.09	0.02
v_{15}	0.09	0.07	0.19	0.11	0.54	0.00	0.00	0.07	0.07	0.17	0.01
v_{16}	0.07	0.31	0.10	0.09	0.43	0.00	0.50	0.00	0.05	0.19	0.02
v_{17}	0.09	0.11	0.01	0.39	0.40	0.00	0.00	0.13	0.00	0.00	0.08
v_{18}	0.13	0.22	0.15	0.09	0.41	0.11	0.00	0.07	0.08	0.16	0.01
v_{19}	0.11	0.17	0.09	0.13	0.49	0.11	0.00	0.13	0.00	0.00	0.08
v_{20}	0.09	0.24	0.00	0.27	0.40	0.00	0.00	0.07	0.00	0.00	0.08

typical strong mines signatures characteristics, i.e. hyperbolic shape of the signature with succession of Dg-Hz-Ad states. This can be corroborated by the results reported in table 4:

1. The means of states s_1, s_2, and s_3 (i.e. the Dg, Hz, and Ad state) have the highest component in the D, H, and A dimension, respectively.

2. In the initial model, the matrix A is nearly symmetric with regards to the latency in states s_1 and s_2 (with approximately 2/3 probability) versus the transition to the consequent state (with $\sim 1/3$ probability). After training, however, the model is less likely to stay in s_2 (0.26 probability) and tends to transition rapidly (0.74 probability) to s_3. This can be explained by the strength of the diagonal and anti-diagonal edges of the mine signatures in cluster 5 (refer to figure 13), compared to the horizontal edge.

3. Codes with high emission probability B in state s_1 ($b_1(v_7)$, $b_1(v_{11})$, $b_1(v_{13})$, $b_1(v_{17})$, and $b_1(v_{19})$) have strong D component. The same remark is valid for codes with high emission probability

TABLE 6

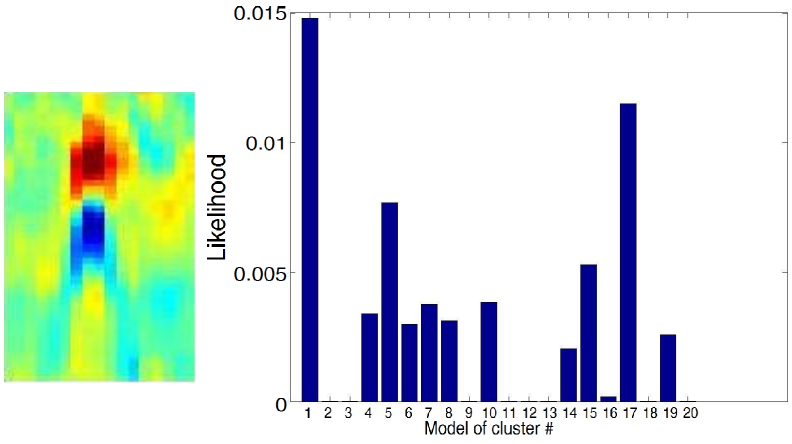
λ_2^F DHMM model parameters of cluster 2

Means						**Initial A**			**A**		
	H	V	D	A	N	s_1	s_2	s_3	s_1	s_2	s_3
s_1	0.14	0.02	0.04	0.04	0.75	0.95	0.05	0.00	0.76	0.24	0.00
s_2	0.17	0.03	0.05	0.06	0.69	0.00	0.85	0.15	0.00	0.31	0.69
s_3	0.14	0.02	0.03	0.05	0.75	0.00	0.00	1.00	0.00	0.00	1.00
Codes						**Initial B**			**B**		
v_1	0.17	0.01	0.00	0.39	0.41	0.00	0.00	0.02	0.00	0.00	0.13
v_2	0.61	0.00	0.00	0.00	0.36	0.00	0.06	0.00	0.03	0.13	0.04
v_3	0.01	0.00	0.00	0.00	0.91	0.09	0.00	0.55	0.04	0.08	0.05
v_4	0.23	0.13	0.17	0.24	0.23	0.00	0.00	0.02	0.00	0.00	0.13
v_5	0.10	0.07	0.07	0.06	0.67	0.13	0.03	0.03	0.10	0.00	0.00
v_6	0.31	0.03	0.03	0.16	0.47	0.01	0.00	0.02	0.00	0.00	0.13
v_7	0.16	0.00	0.01	0.01	0.76	0.36	0.06	0.10	0.05	0.05	0.06
v_8	0.06	0.01	0.06	0.01	0.81	0.17	0.00	0.15	0.10	0.00	0.00
v_9	0.20	0.17	0.06	0.24	0.33	0.00	0.00	0.03	0.00	0.00	0.13
v_{10}	0.29	0.00	0.01	0.01	0.66	0.14	0.35	0.02	0.03	0.16	0.04
v_{11}	0.19	0.14	0.11	0.10	0.43	0.01	0.08	0.01	0.10	0.00	0.00
v_{12}	0.24	0.04	0.08	0.08	0.56	0.03	0.16	0.00	0.03	0.17	0.03
v_{13}	0.36	0.04	0.14	0.14	0.31	0.01	0.03	0.02	0.03	0.14	0.04
v_{14}	0.11	0.24	0.34	0.01	0.20	0.01	0.00	0.00	0.10	0.00	0.00
v_{15}	0.14	0.06	0.64	0.03	0.13	0.00	0.00	0.00	0.10	0.00	0.00
v_{16}	0.46	0.00	0.00	0.01	0.51	0.00	0.08	0.00	0.03	0.14	0.04
v_{17}	0.43	0.01	0.06	0.06	0.46	0.00	0.09	0.01	0.03	0.14	0.04
v_{18}	0.04	0.18	0.19	0.02	0.49	0.01	0.00	0.00	0.10	0.00	0.00
v_{19}	0.16	0.07	0.04	0.20	0.56	0.01	0.05	0.03	0.00	0.00	0.13
v_{20}	0.11	0.07	0.16	0.07	0.54	0.03	0.00	0.00	0.10	0.00	0.00

B in s_2 and state s_3 having strong H and A component, respectively.

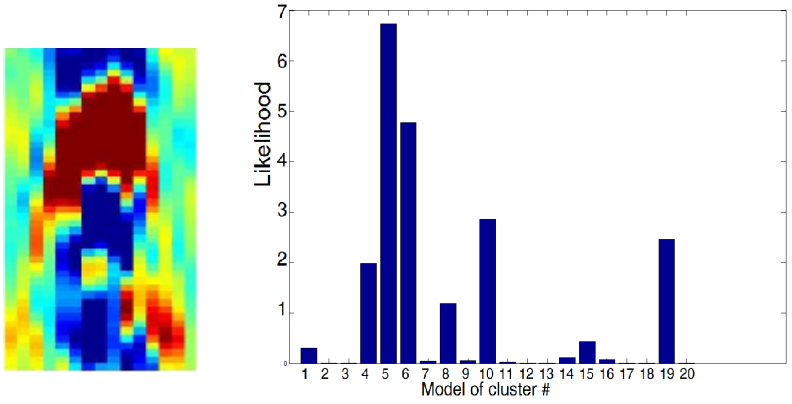
In tables 5 and 6, we show the mine and the clutter model of cluster 2, respectively. As shown earlier in figure 12, cluster 2 has a large number of clutter alarms with few weak mines. Therefore, the means of states s_1, s_2, and s_3 of $\lambda_2^{(M)}$ have respectively weaker D, H, and A components compared to the states means of $\lambda_5^{(M)}$, reported in table 4. However, the non-edge N component of $\lambda_2^{(M)}$ states means ($s_1([N]) = 0.39$, $s_2([N]) = 0.51$, $s_3([N]) = 0.38$) is higher than those of $\lambda_5^{(M)}$ ($s_1([N]) = s_3([N]) = 0.16$ and $s_2([N]) = 0.17$). Moreover, the non-edge component is significantly higher in the $\lambda_2^{(F)}$ model' states means ($s_1([N]) = s_3([N]) = 0.75$ and $s_2([N]) = 0.69$), as shown in table 6.

Given the set $\{\lambda_k^{(c)}, k = 1, \cdots, K, c \in \{M, C\}\}$ of models learned (refer to tables 4, 5, and 6 for few examples of clusters' mine and clutter models), we analyze the performance of these models on the training data. In Figure 16, we plot the likelihood of a mine signature belonging to cluster 1 in all cluster models. As expected, the highest likelihood occurs when testing the sequence with the



(a) (b)

Figure 16. (a) A sample alarm signature assigned to cluster 1. (b) Clusters' models responses to the signature in (a)



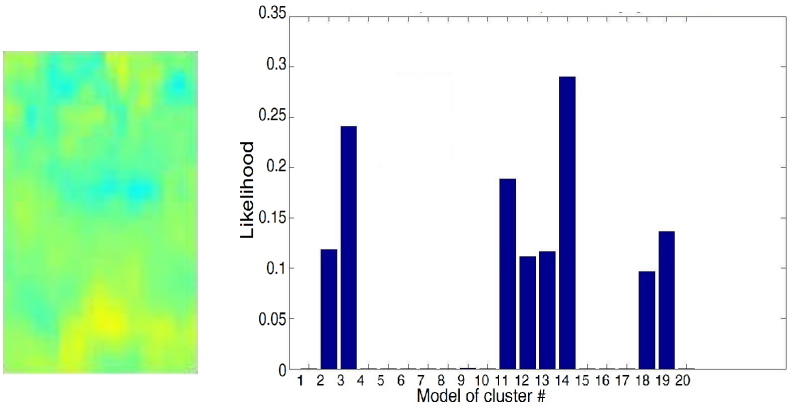
(a) (b)

Figure 17. (a) A sample alarm signature assigned to cluster 5. (b) Clusters' models responses to the signature in (a)

DHMM model of cluster 1. Moreover, the higher likelihood values correspond to the mine-dominated clusters $(1, 4, 5, 6, \cdots, 14, 15, 17, \cdots)$.

The same remark is valid for the models' response to a test signature of a strong mine from cluster 5, as shown in figure 17. In this case, fewer models responded to that signature. These models correspond to clusters containing similar signatures (strong mines). Similarly, figure 18 shows that a test sequence belonging to cluster 2 (which is dominated by clutter signatures) has high likelihood in only clutter-dominated clusters' models.

Figure 19 shows the scatter plot of the log-likelihood of the training data in model 5 (ordinate



(a) (b)

Figure 18. (a) A sample alarm signature assigned to cluster 2. (b) Clusters' models responses to the signature in (a)

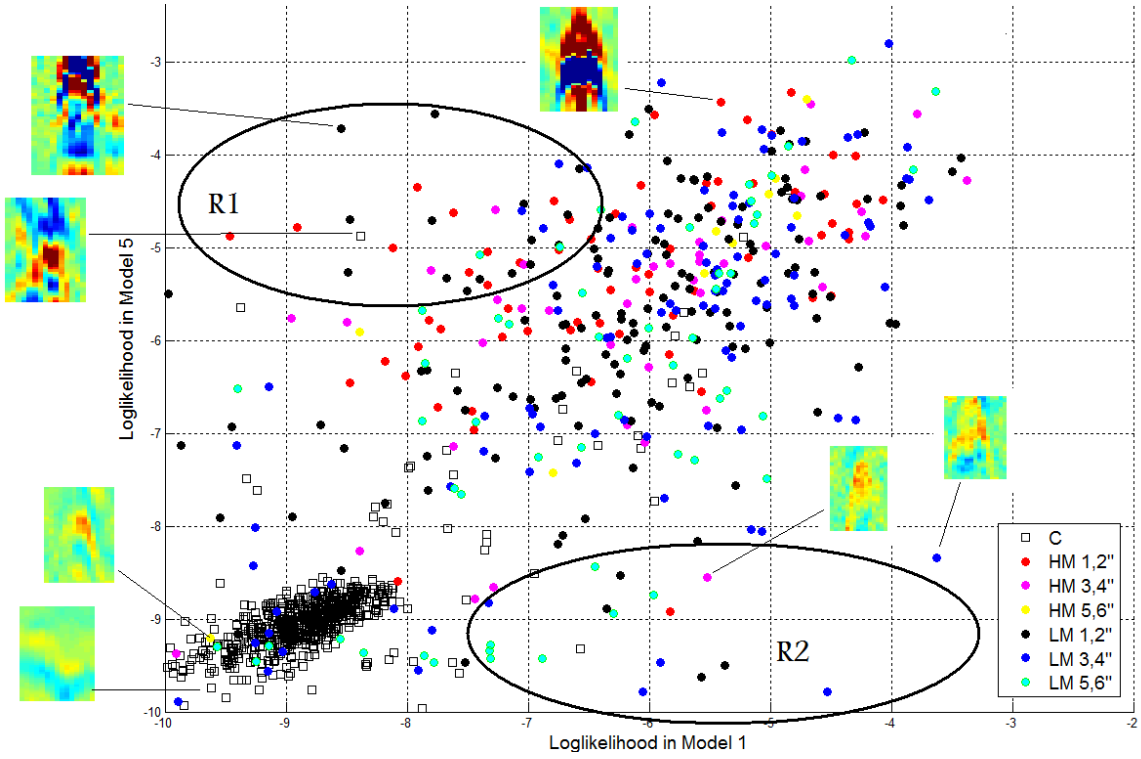


Figure 19. Scatter plot of the log-likelihoods of the training data in model 5 (strong mines) Vs. model 1 (weak mines). Clutter, low metal (LM), and high metal (HM) signatures at different depths are shown with different symbols and colors.

axis) versus model 1 (abscissa axis). In this figure, we display clutter, low metal (LM), and high metal (HM) signatures at different depths using different symbols and colors. Even though the two models are dominated by mine signatures, we see that not all confidence values are highly correlated. In fact, some strong mine signatures have high log-likelihoods in model 5 and lower log-likelihoods in model 1 (upper left side of the scatter plot, region $R1$). This can be attributed to the fact that cluster 5 contains mainly strong mines and is more likely to yield high log-likelihood when testing a strong mine signature. On the other hand, in region $R2$, the performance of cluster 1 model is better as it gave higher likelihood values to the "weak" mines in that region. In fact this result is expected because cluster 1 contains weak mine signatures.

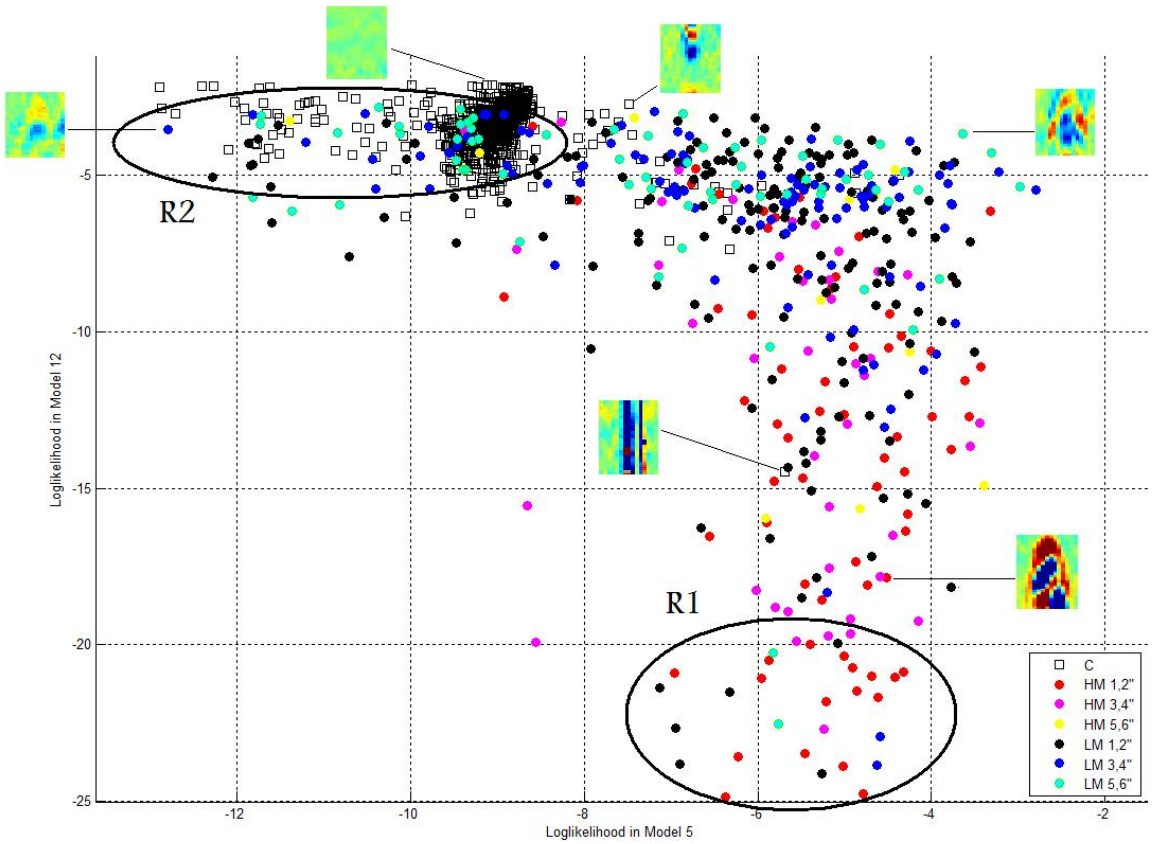


Figure 20. Scatter plot of the log-likelihoods of the training data in model 12 (clutter) Vs. model 5 (strong mines). Clutter, low metal (LM), and high metal (HM) signatures at different depths are shown with different symbols and colors.

In Figure 20, we display the scatter plot of the log-likelihood of all the training data in model 12 (clutter) versus model 5 (mines). As clusters 5 and 12 are dominated respectively by mines and clutter, their results are negatively correlated. That is, cluster 5 assigns higher log-likelihood to mine signatures and lower log-likelihoods to clutter signatures. Cluster 12 model does the opposite. In

region $R1$ in Figure 20, strong signatures (high metal mines and/or mines buried at shallow depths) are being assigned high likelihood values in cluster 5 model and very low likelihood values in cluster 12 model. In region $R2$, a group of clutter signatures and deeply-buried low metal mines have high likelihoods in cluster 12 model and very low likelihood in cluster 5 model.

Figures 19 and 20 illustrate that the learned HMM models give different results on different regions of the input space and that they are potentially complementary.

Although each cluster may have one or two models, its output is a single confidence value. Precisely, the output of the BW-trained models is $Pr(O|\lambda_i)$ while the output of the MCE-trained models is $Pr(O|\lambda_j) = Pr(O|\lambda_j^{(1)}) - Pr(O|\lambda_j^{(0)})$. Therefore, we can consider each cluster model as a classifier and analyze its overall performance on the training data. In figure 21, we plot the ROCs of some cluster models. The clusters ROCs show that the models perform differently at different levels of false alarms rate. We also notice that no model consistently outperforms all other models. For instance, as shown previously in the scatter plots of figures 19 and 20, cluster 5 model assigns high confidence values to strong mines but fails to recognize weak signatures.

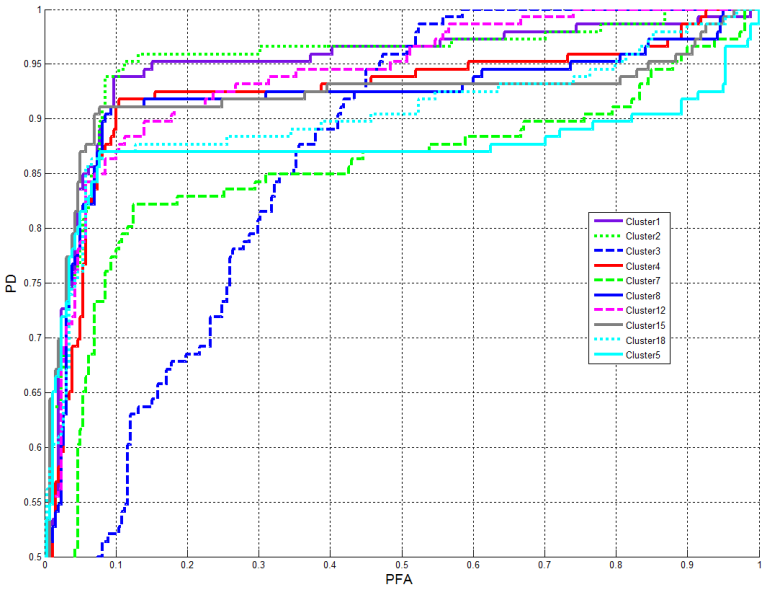


Figure 21. ROCs of some clusters models. Solid line: clusters dominated by mines. Dashed line: clusters dominated by clutter.

3.3.4 Decision level fusion of multiple HMMs

In the previous steps of our ensemble HMM, we initialized and learned multiple models for the different clusters. Any sequence O will be tested by these models. Depending on the type and the contents of the clusters, one or more models may have strong response to the test sequence O. The multiple responses of the different models need to be combined into a single confidence value. We propose using the procedure highlighted in Algorithm 6 to achieve this task.

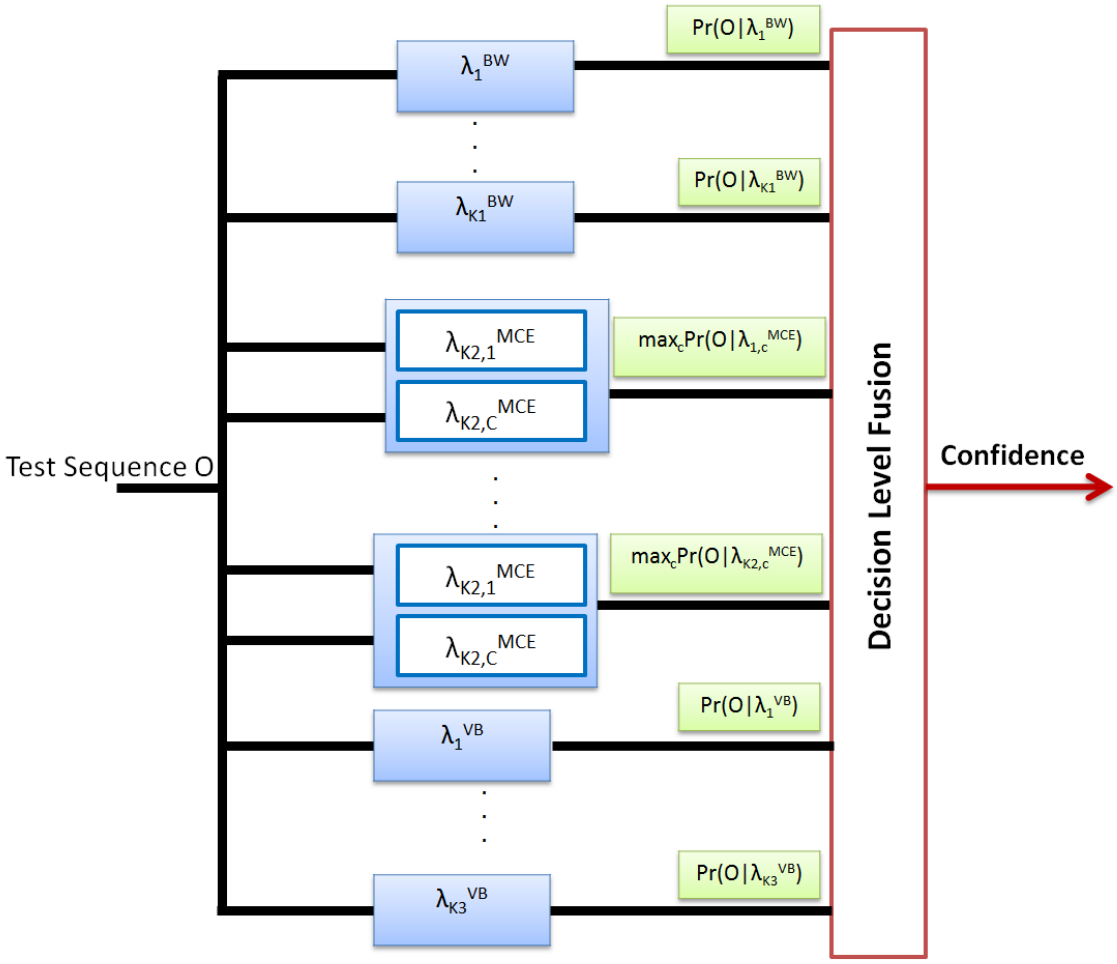


Figure 22. Block diagram of the proposed eHMM (testing)

The block diagram for the proposed eHMM, in testing mode, is given in figure 22. The output of Baum-Welch- and VB-trained cluster models is $Pr(O|\lambda_k)$ while the output of the MCE-trained cluster models is $\max_c Pr(O|\lambda_{k,c}^{MCE})$. Although each cluster may have up to C models, its output is a single confidence value that represents the degree to which the input test sequence "belongs" to that cluster.

Algorithm 6 Testing a new sequence using the eHMM

Require: Test observation O
Ensure:
1: Compute $Pr(O|\lambda_i^{BW})$ for the K_1 clusters learned with BW
2: Compute $Pr(O|\lambda_j^{MCE}) = \max_c Pr(O|\lambda_{j,c}^{MCE})$, for the K_2 clusters learned with MCE
3: Compute $Pr(O|\lambda_k^{VB})$ for the K_3 clusters learned with VB
4: Pr(O) = $\mathbb{H}(Pr(O|\lambda_i^{BW}), Pr(O|\lambda_j^{MCE}), Pr(O|\lambda_k^{VB}))$

The general framework for decision combination is the following. Let $\Lambda = \{\lambda_i^{BW}, \lambda_j^{MCE}, \lambda_k^{VB}\}$ be the resulting mixture model composed of a total of K models, $K = K_1 + K_2 + K_3$. Let

$$\mathbf{F}(k, r) = \log Pr(O_r|\lambda_k), \quad 1 \leq r \leq R, \quad \text{and} \quad 1 \leq k \leq K \tag{92}$$

be the log-likelihood matrix obtained by testing the R training sequences with the K models. Each column $\mathbf{f}_r$ of the matrix $\mathbf{F}$ represents the feature vector of each sequence in the decision space (recall that $\mathbf{f}_r$ is a K-dimensional vector while O_r is a sequence of vector observations of length T, 15×4 sequence in the case of the landmine application example). In fact, each column represents the confidences assigned by the K models to each sequence r. Therefore, the set of sequences $\mathbb{O}= \{O_r, y_r\}_{r=1}^R$ is mapped to an Euclidean confidence space $\{\mathbf{f}_r, y_r\}_{r=1}^R$. Finally, a combination function, $\mathbb{H}$, takes all the $\mathbf{f}_r$'s as input and outputs the final decision.

Several decision level techniques such as simple algebraic [63], artificial neural networks (ANN) [1], and hierarchical mixture of experts (HME) [46] can be used.

3.3.4.1 Algebraic-based fusion

Simple algebraic methods are not trainable and rely on the statistics of the clusters contents and the corresponding models performance. The assumption is that each cluster is associated with one and only one class. A class can be represented by more than one cluster. In the case of MCE-trained cluster models, the corresponding class is the class in which the observation has the maximum likelihood. Let $class(k)$ refer to the class label of the majority of the samples assigned to cluster k. The aggregation function could be a simple majority vote, i.e.,

$$\mathbb{H}(\mathbf{f}_i) = \underset{c=1:C}{\text{argmax}} \sum_{k=1}^{K} f_{ik}\delta(class(k) = c), \quad 1 \leq i \leq R, \tag{93}$$

or the max operator,

$$\mathbb{H}(\mathbf{f}_i) = class(\underset{k=1:K}{\text{argmax}}\, f_{ik}), \quad 1 \leq i \leq R. \tag{94}$$

Since we adopted an unsupervised, non-trainable, approach to data partitioning, it is not always possible to identify reliable cluster-to-class associations and derive adequate statistics. An alternative

is to use ANNs and HMEs to automate these associations and learn the corresponding weights from the performance of the models on the training data.

3.3.4.2 ANN-based fusion

The basic decision combination neural network (DCNN) [1] is a single layer perceptron with no hidden layers taking $\{\mathbf{f}_1, \ldots, \mathbf{f}_R\}$ as input and outputting the predicted labels $\{z_1, \ldots, z_R\}$. Borrowing the framework introduced in [1], the proposed ANN architecture is illustrated in figure 23. The combination function for this network is

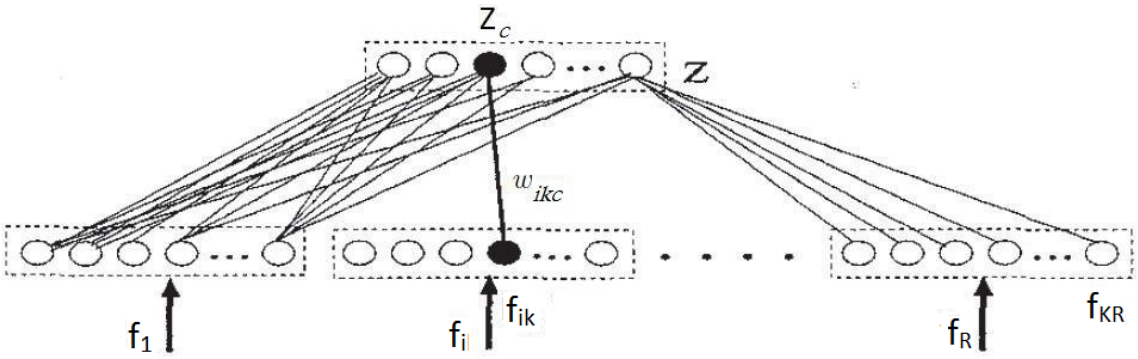


Figure 23. The structure of the basic decision combination neural network (DCNN), adapted from [1]

$$h_c(\mathbf{f}_1, \cdots, \mathbf{f}_R) = \sum_{i=1}^{R} \sum_{k=1}^{K} w_{ikc} f_{ik} + \theta_c, \tag{95}$$

where w_{ikc} and θ_c are respectively the weights and the per-class bias of the DCNN. The final output is casted as a sigmoid function of the form

$$\mathbb{H}_c(\mathbf{f}_1, \cdots, \mathbf{f}_R) = \frac{1}{1 + e^{-h_c(\mathbf{f}_1, \cdots, \mathbf{f}_R)}}. \tag{96}$$

The advantage of this architecture is that each weight has an interpretation for the role it plays in the decision combination. Weight w_{ikc} is the contribution to class C_c when classifier λ_k assigns confidence f_{ik} to data sample i. The weights are learned using a backpropagation [64] algorithm that minimizes the mean square error (MSE) between the true labels and the actual outputs of the DCNN on the training data.

The DCNN framework has several desirable properties. First, it can be considered as a generalization of the linear combination models. Second, it is easily trainable. In other words, training data could be used to learn the weights. Finally, it is not sensitive to possibly correlated classifiers, as redundant classifiers could be eliminated through standard ANN pruning techniques

[65]. The main drawback of ANNs is that they are not easily interpretable and could suffer from overfitting.

3.3.4.3 HME-based fusion

As highlighted in section 2.3.3, the HME method utilizes a tree-based hierarchy of experts and can be applied to various supervised learning problems. For the eHMM framework, models learned from clusters of observations can be treated as experts in different domains. The leaves of such a hierarchy represent experts, while the non-leaves constitute a gating network designed to dynamically combine the outputs of the subtrees. The input to the HME network is a K-dimensional vector $\mathbf{F}$. Equation (64) maps the input of the HME to the output, denoted y, via a probabilistic model. In this case, $\mathbb{H}$ is a probability density function, P, conditioned on the input $\mathbf{F}$ and the model parameters θ of the HME:

$$P(y|\mathbf{F},\theta) = \sum_i g_i \sum_j g_{j|i} P(y|\mathbf{F},\theta_{ij}). \tag{97}$$

In the case of binary classification, the output y is a discrete random variable having two possible classes (or two possible values: 0 or 1). In this case, equation (97) reduces to:

$$P(y|\mathbf{F},\theta) = \sum_i g_i \sum_j g_{j|i} \mu_{ij}^y (1-\mu_{ij})^{1-y}. \tag{98}$$

Here μ_{ij} is the conditional probability of classifying the input as class 1. Note that the dependence on the input is through the gating networks g_i and $g_{j|i}$.

The HME parameters are learned from data via an EM-like algorithm [46] or via gradient descent techniques [45].

3.3.4.4 Illustrative example: fusion results using HME and ANN

In our illustration example, we combine the cluster model outputs using two methods: ANN and HME fusions. In the case of ANN combination, the weights are relative to the clusters models' performance and are learned from the models' response to the training data. In figure 24, we report the weights assigned to each cluster model after training the ANN. Typically, one should expect that the weights assigned to clutter dominated models are negative and those assigned to mine dominated clusters are positive.

In the case of HME combination, each model is considered an "expert" in a particular region of the feature space. The gating networks of the HME control the regions on which a particular

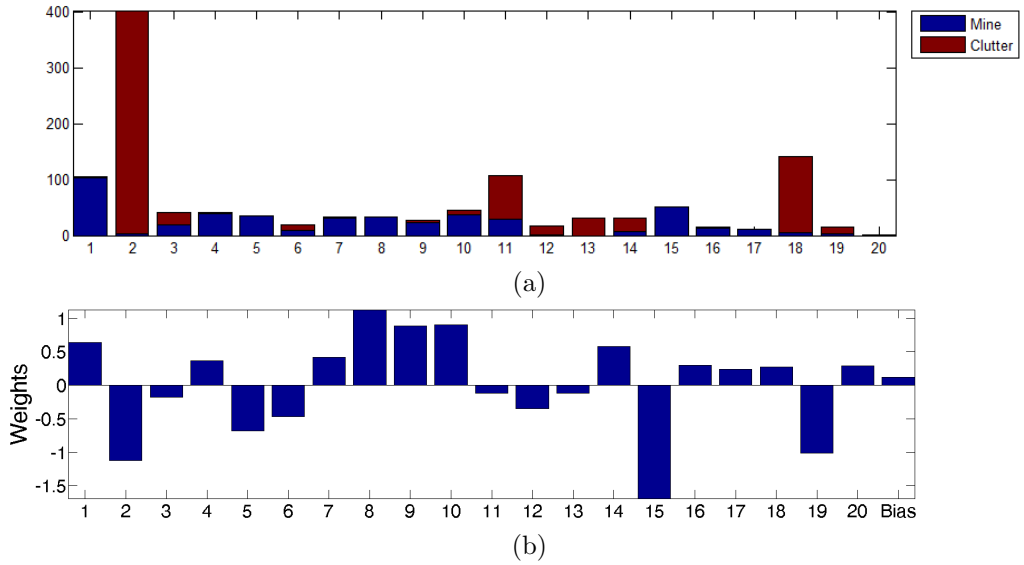


Figure 24. Weights assigned to the 20 HMMs using ANN. (a) Distribution of the alarms in each cluster per class (mines vs. clutter). (b) ANN weights and bias

expert model has better performance. The posterior probability at a particular node in the HME is the likelihood output of the expert model. The gating network coefficients are learned by optimizing the performance of the HME network on the training data using the EM algorithm. We use a 1-level HME with branching factor of two. We show the gating network parameters associated to each model (expert network) in table 7. The gating network parameters , v_i's, can be viewed as a soft partition of the original $K-$dimensional confidence space.

Given a test observation x, its inner product with the gating networks vectors (shown in table 7), combined with the softmax smoothing function, determines the degree to which the test point is to be treated by each of the expert networks. Geometrically, vectors $\mathbf{v_0}$, $\mathbf{v_{11}}$, and $\mathbf{v_{12}}$ can be viewed as the normals to the regression surfaces of the gating networks. The higher the absolute value of one covariate of $\mathbf{v}$, the more weight is given to the corresponding model confidence. As mentioned earlier, the parameters $\mathbf{v_0}$, $\mathbf{v_{11}}$, and $\mathbf{v_{12}}$ are learned via the EM algorithm. In figure 25, we show the evolution of the mean log likelihood along the EM iterations. As it can be seen, the overall log-likelihood of the HME reaches a plateau around iteration #40.

To evaluate the HME and ANN fusion methods, we compare the ROCs generated by these two methods, using the training data, to the ROCs of the best three clusters' models. As it can be seen in figure 26, both fusion methods are consistently better than the three clusters' models. Also, we can see that the HME fusion slightly outperforms the ANN fusion. This is possibly due to the fact that the HME is more suited for highly non linear classification problems. In this kind

TABLE 7

HME gating network parameters after EM training

Dimension	$\mathbf{v_0}$	$\mathbf{v_{11}}$	$\mathbf{v_{12}}$
1	-170.222	-0.355	-0.015
2	48.345	-0.498	0.485
3	-479.007	-0.632	-2.609
4	13.281	-0.264	0.351
5	51.229	0.220	-1.485
6	-129.177	0.188	-1.712
7	102.388	0.564	1.402
8	180.394	1.243	1.212
9	249.114	0.882	2.778
10	29.622	-0.023	0.354
11	175.361	-1.182	3.203
12	4.915	-0.191	-1.015
13	-96.080	0.072	-0.644
14	-135.543	-1.182	2.518
15	57.147	-0.447	0.318
16	-55.869	-0.276	-0.421
17	43.716	0.209	0.504
18	-9.176	0.429	-2.111
19	-531.275	-0.888	-2.791
20	36.375	0.075	0.219
Intercept	-154.275	0.087	14.588

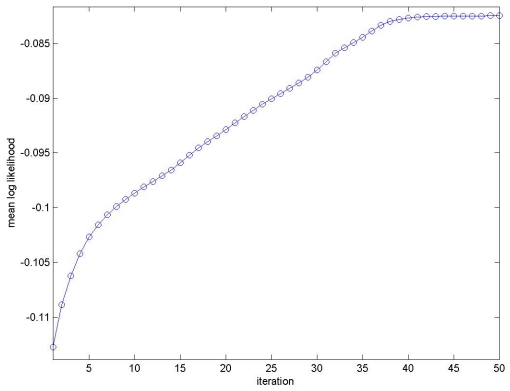


Figure 25. Log-likelihood of the HME during EM training

of settings (high non linearity), the ANN is prone to overfitting while the HME addresses the high nonlinearity through the "soft" partitioning of the input space, i.e. the divide and conquer approach.

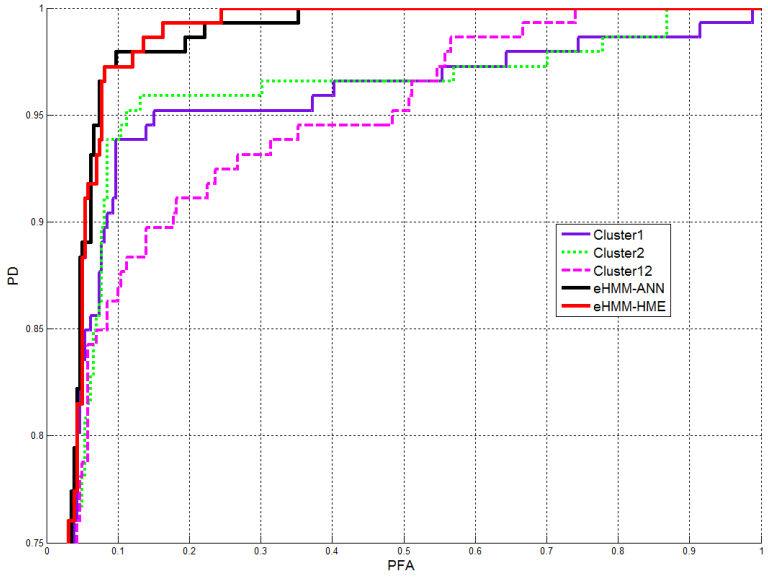


Figure 26. Comparison of the ANN and HME fusion with the best three cluster models (1, 2, and 12).

3.4 Chapter summary

In this chapter, we proposed our ensemble HMM classifier and its components. First, We gave the motivations behind adopting our approach. Then, we detailed the different components/steps of the proposed ensemble HMM classifier, namely, the similarity matrix computation, the pairwise-similarity-based clustering, the models initialization and training, and the decision level fusion. We have used an example of landmine detection using GPR signatures and EHD features to illustrate and motivate each step. In the next chapters, we provide more details about the application of the ensemble HMM classifier to real world sequential data sets. In particular, we analyze, in chapter 4, the results of the eHMM application to identify CPR scenes in video simulating medical crises. Then, in chapter 5, we provide a more comprehensive results and analysis of the application of eHMM to landmine detection.

CHAPTER 4

APPLICATION TO CPR SCENES IDENTIFICATION

In this chapter, the proposed ensemble HMM classifier is used to identify CPR scenes in video simulating medical crises. First, we provide the steps needed for extracting motion-based features from the training videos. Then, we outline the proposed eHMM classifier architecture. Finally, we compare the eHMM classifier's performance to the baseline HMM classifier.

4.1 Introduction

Medical simulations, where uncommon clinical situations can be replicated, have proved to provide a more comprehensive training. Simulations involve the use of patient simulators, which are lifelike mannequins that have respiration, heartbeat, and respond to treatment with virtual drugs. Simulations sessions involve 4 to 9 people and last approximately 30 minutes to one hour. Simulation training sessions are scheduled approximately twice per week and are recorded as video data. After each session, the physician/instructor must manually review and annotate the recording and then debrief the trainees on the session. Video-assisted debriefing allows participants to de-construct and reflect on their experiences, teaching them how to approach such tasks more effectively in the future.

The physician responsible for the simulation sessions has recorded over 100 sessions, and is now realizing that: (1) the manual process of review and annotation is labor intensive; (2) retrieval of specific video segments is not trivial; and (3) there is wealth of information waiting to be mined form these recordings. Providing the physician with automated tools to segment, semantically index and retrieve specific scenes from a large database of training sessions will enable him/her to: (1) immediately review important sections of the training with the team; (2) allow more efficient debriefing session with the team of trainees; and (3) identify similar circumstances in previously recorded sessions. The longer term payback is the potential discovery of similar critical elements in a training session that result in either positive or negative outcomes and thus enhance the effectiveness of the training.

4.2 Identification of CPR scenes in video simulating medical crises

In this section, we outline our approach to use the eHMM approach to detect and classify scenes that involve CPR activity. Our approach consists of two main steps. The first one segments the video into shots, selects one keyframe for each shot, and identifies regions with skin-like colors in each keyframe. Each skin region is then represented by a sequence of observations that encode its motion in the different frames within the shot boundaries. The second step consists of building an ensemble HMM classifier that uses motion-based features in order to identify the skin-like regions that involve CPR activities.

4.2.1 Video and image segmentation

First, a process that extracts shot boundaries to make a set of shots is needed. A shot is defined as a video segment within a continuous capture period. To include the effect of dynamic visual content, various cues such as color, motion, mosaic of frames can be adopted. The most common approaches to shot boundary detection are based on color histogram [66], probability theory [67], and unsupervised clustering [68]. Other approaches, based on motion are generally better suited for controlling the number of frames based on temporal dynamics in the scene. These approaches include pixel-based image differences [69], optical flow computation [70], and global motion and gesture analysis [71].

After shot boundary detection, a key-frame that reflects the main content of each scene needs to be extracted. Most of the early work selects key-frames by randomly or uniformly sampling the video frames from the original sequence at certain time intervals [72]. Since a shot is defined as a video segment within a continuous capture period, a natural and straightforward way of key-frame extraction is to use the first frame of each shot as its key-frame [73, 70]. In our proposed approach, we adopt the latter method. Next, each key-frame is segmented into a small number of homogeneous regions. These regions will provide an organization framework for subsequent scene analysis. Segmentation is achieved by clustering the color pixel information into non overlapping regions of similar features. Any clustering algorithm could be used to achieve this task. We use the normalized cut algorithm (Ncut) [2]. Our choice is motivated by the computational efficiency of this algorithm and its ability to cluster the image into a reasonable number of regions with no supervision information. Figure 27(a) displays a key-frame from a sample shot extracted from one of the training sessions. Figure 27(b) displays the segmented regions obtained by using Ncut.

(a) Key-frame of a sample shot (b) Segmented key-frame

Figure 27. Key-frame segmentation using Ncut [2] algorithm

4.2.2 Skin detection

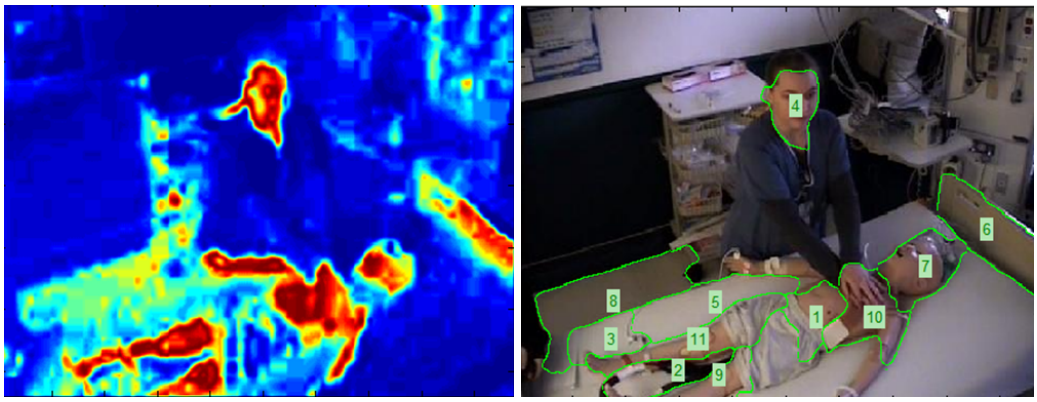
Instead of processing all detected regions, we identify only those that are of interest. Since our objective consists of identifying CPR scenes and since this action typically involves the trainees hands and the mannequin chest, we identify and keep only those regions with skin-like colors. To achieve this task, we use a simple but efficient skin pixel classifier [74] to discriminate between skin and non-skin regions. We use a collection of skin images to train the classifier. This collection includes 500 images of hands at different orientations and under different illumination. First, each image is mapped to the (r, g) color space where $r = \frac{R}{(R+G+B)}$ and $g = \frac{G}{(R+G+B)}$. This normalized color space has proved to be suitable for representing skin color under different lighting conditions. Each image is then filtered using a low-pass filter to reduce the effect of noise. Second, the color distribution of all pixels in all images is fitted by a Gaussian model with mean μ and a covariance matrix Σ.

Let $x_{ij} = (r_{ij}, g_{ij})$ be the color of the i^{th} pixel in region R_j. The likelihood of x_{ij} in the Gaussian skin model can be estimated using

$$\mathfrak{L}(x_{ij}) = exp(-0.5(x_{ij} - \mu)\Sigma^{-1}(x_{ij} - \mu) \tag{99}$$

A region is labeled as a skin region, and retained for further processing, if many of its pixels have likelihood larger than a given threshold θ.

Figure 28 illustrates the skin detection process. In figure 28(a), we display the likelihood of all pixels of figure 27(a) in the Gaussian skin model. Here, red pixels indicate high likelihood while blue pixels indicate zero likelihood.



(a) Skin likelihood (b) Identified skin regions

Figure 28. Skin detection using a Gaussian skin model

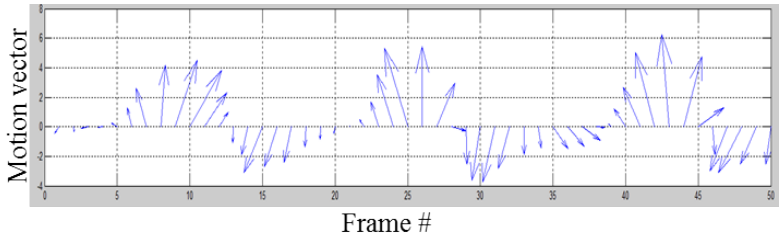
(a) One of the skin regions (b) Optical flow

Figure 29. Optical flow of one of the identified regions in figure 28(b)

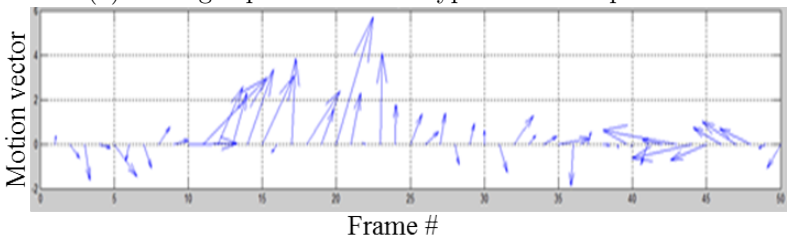
4.2.3 Motion features extraction

Motion is an efficient feature to analyze moving objects that change location with time [75]. For each skin region, we extract a motion feature vector. First, we compute the optical flow for all pixels within the region using Horn-Schunck algorithm [76]. Optical flow measures the change in velocity in terms of speed and direction at each pixel location. Then, the optical flow, $\bar{\mathbf{u}}_j = (\bar{u}_j^x, \bar{u}_j^y)$, of the region of interest R_j, is estimated as the average of the optical flow of all of its pixels. Figure 29(a) displays one of the skin labeled regions in figure 28(b). In figure 29(b), we superimpose the optical flow of each pixel in the region as well as the average optic flow $\bar{\mathbf{u}}_j$ (in red). In this case, the pixels are moving downward, indicating the push-down phase of the CPR.

The average optical flow $\bar{\mathbf{u}}_j$ is computed at each frame of a given shot. Thus, skin region R_j



(a) Average optical flow of a typical CPR sequence



(b) Average optical flow of a non-CPR sequence

Figure 30. Average optical flow sequence of two sample regions

within the k^{th} video shot will be represented by the sequence of observations:

$$\bar{\mathbf{O}}_{jk} = \{\bar{\mathbf{u}}_{jk}^{1}, \cdots, \bar{\mathbf{u}}_{jk}^{t}, \cdots, \bar{\mathbf{u}}_{jk}^{T}\}, \tag{100}$$

where T is the number of consecutive frames, within video shot k, that include skin region R_j. Figure 30 displays the average optical flow of one skin region that involves CPR activities and another non-CPR skin region over a sequence of 50 frames. As it can be seen, the CPR sequence can be characterized by an upward motion followed by a downward motion. On the other hand, the non-CPR sequence has no specific pattern and the motion vector tend to be random.

4.2.4 Video shot classification using HMMs

4.2.4.1 Baseline HMM classifier

To discriminate between skin regions that are involved in CPR activities and other skin regions, we developed and trained a Hidden Markov Model (HMM) classifier that is based on motion features. The baseline HMM classifier (bHMM) is comprised of two models: one for CPR sequences and one for non-CPR sequences. Each model has two states and produces a probability value by backtracking through model states using the Viterbi algorithm [8]. The CPR model, λ^{CPR}, is designed to capture the smooth transition between state 1 and state 2 that characterizes CPR sequences. The non CPR model, $\lambda^{\sim CPR}$, is needed to capture the characteristics of non-CPR sequences. The probability value produced by the CPR (non-CPR) model can be thought of as

an estimate of the probability of the observation sequence given that there is a CPR (non-CPR) model present. These probabilities are produced by backtracking through the model states using the Viterbi algorithm [8] . The confidence value assigned to each observation sequence, $Conf(O)$, depends on: (1) the probability assigned by the CPR model λ^{CPR}, and (2) the probability assigned by the non-CPR model $\lambda^{\sim CPR}$. In particular, we use:

$$Conf(O) = \log Pr(O|\lambda^{CPR}) - \log Pr(O|\lambda^{\sim CPR}). \tag{101}$$

4.2.4.2 Ensemble HMM classifier

In the baseline HMM classifier, the CPR scenes are modeled by a single HMM. The goal is to generalize from all the training data in order to classify unseen scenes. However, generalizing from all the motion vectors might lead to too much averaging and thus loosing some of the discriminating characteristics of the CPR scenes. An illustrative example of this drawback was given in section 3.2. In order to overcome this limitation, the two model classifier is replaced by our ensemble HMM (eHMM). Using multiple models can capture the variations of the CPR scenes that may be lost under the effect of averaging in the two models case. The implementation of the eHMM for CPR classification will be detailed later in section 4.3.4.

4.3 Experimental results

4.3.1 Data collection and preprocessing

To validate our proposed approach to detect CPR shots, we use the video of one simulation session. The duration of the session is 30 min and 19 sec, with 29 frames per sec. Each frame has 720×480 resolution. After shot boundary detection, key-frame segmentation, and skin detection, we obtain a total of $N = 122$ skin regions. We track each region, R_j, over all frames within its shot (k) boundary, and extract the sequence of observations O_{jk} (as defined in (100)). For validation purposes, we examine each sequence and assign the ground truth labels (CPR or non-CPR). We obtain a total of 42 sequences labeled positively as representing CPR activity, and 80 non-CPR sequences. We also fix the sequence length T to 20 even though HMM can handle sequences of variable length. This is a reasonable assumption as 20 sequences tend to cover the upward and downward motion of the hands in most CPR sequences. Moreover, HMM classifiers are simpler and more efficient when all sequences have a constant length.

For our experiments, we use a 4-fold cross validation. For each fold, a subset of the data is

used for training ($\mathfrak{D}_{Trn}$) and the remaining data is used for testing ($\mathfrak{D}_{Tst}$). Let R_n be the size of $\mathfrak{D}_{Trn}$ and $Q_n = N - R_n$ be the size of $\mathfrak{D}_{Tst}$ for the n^{th} fold. For training, we denote ($\mathfrak{D}_{Trn}^{CPR}$) and ($\mathfrak{D}_{Trn}^{\sim CPR}$) the subsets of the CPR sequences and the non-CPR sequences, respectively.

4.3.2 Evaluation: Receiver Operating Characteristic (ROC) curve

The ROC curve, is a graphical plot of the sensitivity vs. specificity for a binary classifier system as its discrimination threshold is varied. The ROC can also be represented equivalently by plotting the fraction of true positives (TPR = true positive rate) vs. the fraction of false positives (FPR = false positive rate). Consider a two-class prediction problem (binary classification), in which the outcomes are labeled either as positive (p) or negative (n) class. There are four possible outcomes from a binary classifier. If the outcome from a prediction is p and the actual value is also p, then it is called a true positive (TP); however if the actual value is n then it is said a false positive (FP). Conversely, a true negative occurs when both the prediction outcome and the actual value are n, and false negative is when the prediction outcome is n while the actual value is p. Let us define an experiment from P positive instances and N negative instances. The four outcomes can be formulated in a 2×2 contingency table or a confusion matrix, (refer to table 8). To draw an ROC curve, only the true positive rate (TPR) and false positive rate (FPR) are needed. TPR determines a classifier or a diagnostic test performance on classifying positive instances correctly among all positive samples available during the test. FPR, on the other hand, defines how many incorrect positive results while they are actually negative among all negative samples available during the test.

An ROC space is defined by FPR and TPR as x and y axes respectively, which depicts relative trade-offs between true positive (benefits) and false positive (costs). Since TPR is equivalent with sensitivity and FPR is equal to 1-specificity, the ROC graph is sometimes called the sensitivity vs (1-specificity) plot. Each prediction result or one instance of a confusion matrix represents one point on the ROC curve.

TABLE 8

Contingency table

	p	n	total
p'	True Positive	False Positive	P'
n'	False Negative	True Negative	N'
total	P	N	P+N

4.3.3 Video shot classification using baseline HMM

For each cross validation fold, we build a two-model classifier: one HMM model, λ^{CPR}, based on the subset ($\mathfrak{D}_{Trn}^{CPR}$) of CPR training sequences; and another model, $\lambda^{\sim CPR}$, based on non-CPR training sequences ($\mathfrak{D}_{Trn}^{\sim CPR}$). We assume that λ^{CPR} has two states s_1 and s_2: one represents the upward motion of the hands and the second represents the downward motion. These states are estimated using:

$$\begin{cases} s_1 = average\{\bar{\mathbf{u}}_j \in (\mathfrak{D}_{Trn}^{CPR}) | \bar{\mathbf{u}}_j^y \geq 0\} \\ s_2 = average\{\bar{\mathbf{u}}_j \in (\mathfrak{D}_{Trn}^{CPR}) | \bar{\mathbf{u}}_j^y < 0\}. \end{cases} \tag{102}$$

The non-CPR HMM model, $\lambda^{\sim CPR}$, also has two states. Since this model follows no specific pattern, we let its states be the two centers obtained by clustering $\mathfrak{D}_{Trn}^{\sim CPR}$ using the Fuzzy C-Means (FCM) algorithm [57] with C=2.

The remaining parameters $(\mathbf{A}, \mathbf{B}, \pi)$ of λ^{CPR} and $\lambda^{\sim CPR}$ are initialized as follows. The priors, π_1 and π_2, are estimated as the percentage of training sequences that start with state s_1 and s_2 respectively. The transition matrix is initialized using:

$$A = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}.$$

The initialization of the emission probabilities for each state depends whether we use a discrete or a continuous HMM model.

- For the discrete case, the emission matrix, B, is initialized based on the states and a codebook of size M=20. First, codewords $\{\mathbf{v}_m, m = 1, \cdots, 20\}$ are estimated as the centers obtained by clustering the training data using the FCM algorithm with $C = 20$. For the CPR model, we eliminate codewords that have low magnitude ($|\bar{\mathbf{u}}^y| < 0.5$). This will prevent observations with small motion vectors (mainly from non-CPR sequences) from having high probabilities in those codewords. Then, the emission probabilities are initialized using

$$b_j(m) = \left[\sum_{i=1}^{2} \frac{\left\| \mathbf{v}_m - s_j \right\|}{\left\| \mathbf{v}_m - s_i \right\|} \right]^{-1}, \quad for \quad j = 1, 2 \quad and \quad m = 1, \cdots, 20. \tag{103}$$

  Finally, $b_j(m)$ are normalized such that the probabilities in each state sum to one.

- For the continuous case, the emission probabilities are modeled by mixtures of Gaussians with three components for each state. For each state, we cluster the observations belonging to that state using FCM algorithm with $C = 3$. The means of the Gaussian mixture components of

each state are the centers of the clusters. The covariance of each state component is estimated using the observations that belong to the corresponding cluster. For computational reasons, only the diagonal elements of the covariance are considered and all covariance values are lower-bounded to a threshold $MinVar = 0.1$.

After initialization, the transition probabilities and the emission probability parameters of each model are fine-tuned using either the discrete or continuous version of the Baum-Welch learning algorithm [9] and the respective training data. We denote λ_{DHMM}^{CPR} and λ_{CHMM}^{CPR} the resulting discrete and continuous BW-trained CPR models, respectively. Similarly, we define $\lambda_{DHMM}^{\sim CPR}$ and $\lambda_{CHMM}^{\sim CPR}$ for the non-CPR models.

4.3.3.1 Baseline discrete HMM results

TABLE 9

Baseline DHMM model parameters of λ_{DHMM}^{CPR}

Means			**Initial A**		**A**	
	$\bar{u}^x$	$\bar{u}^y$	s_1	s_2	s_1	s_2
s_1	0.16	0.64	0.5	0.5	0.85	0.15
s_2	-0.11	-0.48	0.5	0.5	0.13	0.87
Codes			**Initial B**		**B**	
v_1	-5.47	-1.65	0.00	0.00	0.00	0.00
v_2	2.63	1.77	0.00	0.00	0.00	0.01
v_3	0.54	1.75	0.00	0.00	0.05	0.00
v_4	0.28	1.09	0.41	0.00	0.16	0.00
v_5	-0.22	-0.96	0.00	1.00	0.09	0.88
v_6	-0.65	-1.21	0.00	0.00	0.00	0.11
v_7	-0.23	1.36	0.00	0.00	0.06	0.00
v_8	-0.21	0.82	0.59	0.00	0.57	0.00
v_9	1.03	1.21	0.00	0.00	0.06	0.00

Table 9 displays the baseline DHMM model λ_{DHMM}^{CPR} parameters. In particular, the state means, state transition probabilities, the codewords, and their emission probabilities in each state are shown for both the initial and the Baum-Welch trained models. Since codewords that have low magnitude ($|\bar{\mathbf{u}}^y| < 0.5$) were discarded, λ_{DHMM}^{CPR} has only nine codewords out of the initial 20 obtained by the FCM clustering. Precisely, codewords $\{v_2, v_3, v_4, v_7, v_8, v_9\}$ have positive $\bar{\mathbf{u}}^y$ component, and $\{v_1, v_4, v_5\}$ have negative $\bar{\mathbf{u}}^y$ motion vector component. The corresponding probability of emission of each codeword does not depend on $\bar{\mathbf{u}}^y$ but rather on the distance of the codeword to the mean of each state. Moreover, the normalization step of B matrix rows allows for codewords with negative (resp. positive) $\bar{\mathbf{u}}^y$ to have higher probability in state 1 (resp. state 2). This can be

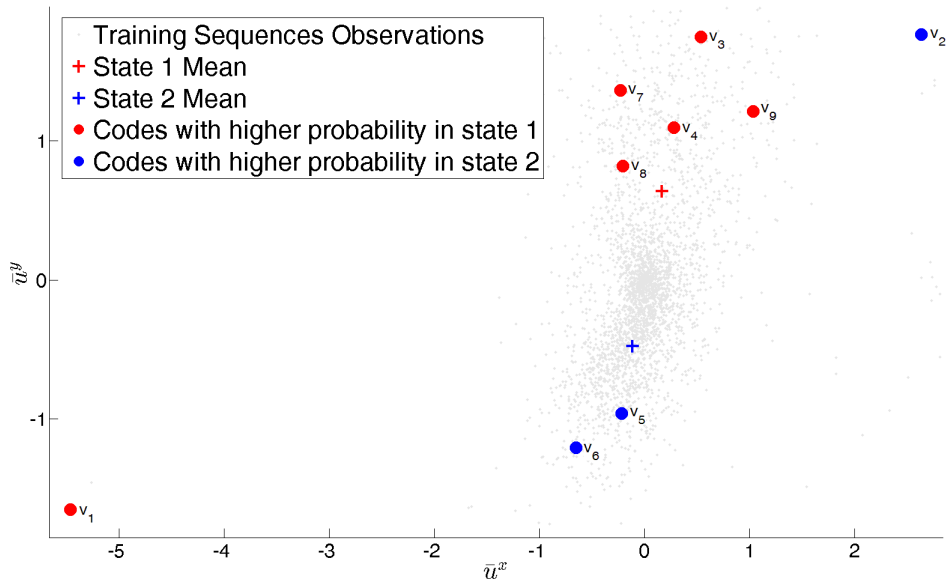


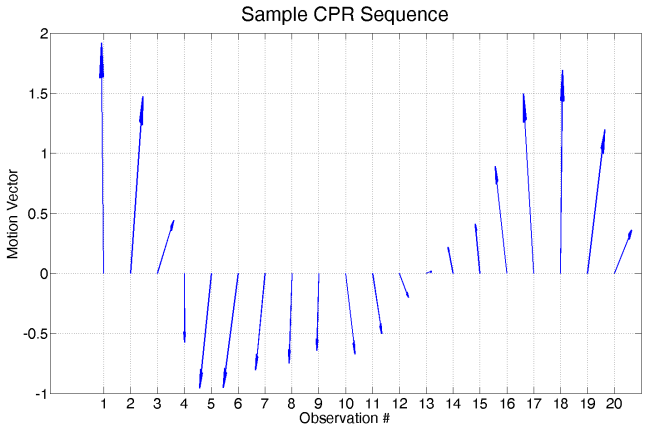
Figure 31. Training observations, states representatives, and codewords of λ^{CPR}_{DHMM}

illustrated better by visualizing λ^{CPR}_{DHMM} parameters in the ($\bar{\mathbf{u}}^x$,$\bar{\mathbf{u}}^y$) plane (refer to figure 31).

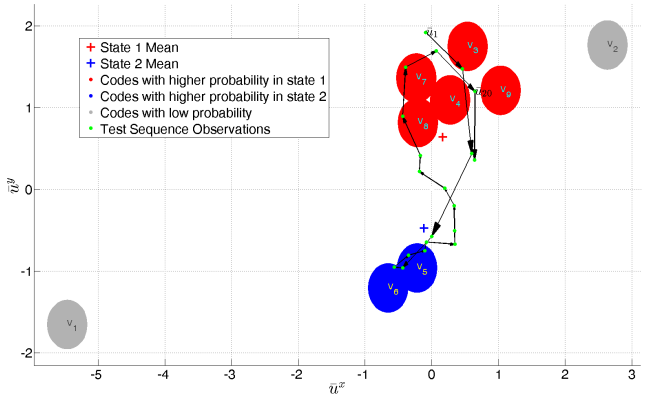
Figure 31 displays the observation vectors used to train the CPR model λ^{CPR}_{DHMM} (first cross validation set). In this figure, we also display the states means and the codewords obtained by clustering the observations. As mentioned earlier, we notice that the codeword v_1 (resp. v_2) has higher probability in state 1 (resp. state 2) even though its $\bar{\mathbf{u}}^y$ value is negative (resp. positive). However, both v_1 and v_2 have low probabilities in both states (≤ 0.01 as shown in table 9). Thus, v_1 and v_2 will not contribute significantly in the test a new sequence in λ^{CPR}_{DHMM}.

In table 10, we display the initial and the Baum-Welch trained $\lambda^{\sim CPR}_{DHMM}$ parameters. We notice that the diagonal elements of the transition matrix A are higher than those of the CPR model of table 9. This means that non-CPR sequences are more likely to stay in one of the states, unlike the CPR sequences that reflect an alternation of upward/downward motion along $\bar{\mathbf{u}}^y$ direction. We notice also that all codewords of $\lambda^{\sim CPR}_{DHMM}$ with high magnitude $\bar{\mathbf{u}}^y > 1$, have a low emission probability (≤ 0.01) in both states (e.g. $v_1, v_2, v_3, v_8, \cdots$).

Figure 32 illustrates the testing process of a sequence by λ^{CPR}_{DHMM}. In Figure 32(a), we display the 20 observations of a typical CPR sequence. As it can be seen, this sequence corresponds to a region that starts with an upward motion (3 observations), followed by downward motion (9 observations), and then back to upward motion (8 observations). In Figure 32(b), we display (in green dots) the same sequence in the ($\bar{\mathbf{u}}^x$; $\bar{\mathbf{u}}^y$) plane. As it can be seen, the first and last



(a) 20 observation vectors of a typical CPR sequence



(b) Path assigned by the Viterbi algorithm for the test sequence in (a)

Figure 32. Illustration of testing a sequence with λ^{CPR}_{DHMM}

observations (upward motion) will be assigned to the codes with higher probability in state 1 while the middle observations (downward motion) will be assigned to the codes with higher probability in state 2. The optimal state sequence, assigned by the Viterbi algorithm, to this test sample is **11122222222211111111**

To evaluate the proposed approach, we score it in terms of probability of detection (PD) versus probability of false alarms (PFA). Confidence values are thresholded at different levels to produce the receiver operating characteristics (ROC) curve.

Figure 33 compares the ROC curves generated by the trained λ^{CPR}_{DHMM}, $\lambda^{\sim CPR}_{DHMM}$, λ^{CPR}_{DHMM} - $\lambda^{\sim CPR}_{DHMM}$ models (in solid lines) and the corresponding initial models before training (in dashed lines). As it can be seen, the CPR model outperforms the non-CPR model as the former one uses prior knowledge to initialize and restrict the model parameters. We notice also that the BW-training improves significantly the classification performance of the CPR model and the baseline DHMM

TABLE 10

Baseline DHMM model parameters of $\lambda_{DHMM}^{\sim CPR}$

Means			**Initial A**		**A**	
	$\bar{u}^x$	$\bar{u}^y$	s_1	s_2	s_1	s_2
s_1	0.04	0.24	0.5	0.5	0.87	0.13
s_2	-0.02	-0.18	0.5	0.5	0.08	0.92
Codes			**Initial B**		**B**	
v_1	-0.23	2.47	0.00	0.00	0.01	0.00
v_2	-3.19	1.16	0.00	0.00	0.01	0.00
v_3	-1.19	-3.61	0.00	0.00	0.00	0.00
v_4	0.56	0.28	0.00	0.00	0.15	0.00
v_5	-0.32	-0.20	0.03	0.27	0.00	0.11
v_6	-1.78	-1.03	0.00	0.00	0.01	0.00
v_7	-0.94	0.42	0.00	0.00	0.06	0.00
v_8	-6.65	-4.78	0.00	0.00	0.00	0.00
v_9	0.00	-0.02	0.15	0.25	0.05	0.50
v_{10}	-1.57	0.21	0.00	0.00	0.03	0.00
v_{11}	-0.32	-1.04	0.00	0.00	0.00	0.03
v_{12}	0.09	0.03	0.25	0.18	0.30	0.12
v_{13}	-3.27	-4.78	0.00	0.00	0.00	0.00
v_{14}	2.03	0.71	0.00	0.00	0.01	0.00
v_{15}	0.41	-0.25	0.00	0.14	0.00	0.11
v_{16}	-5.52	4.88	0.00	0.00	0.00	0.00
v_{17}	0.08	9.62	0.00	0.00	0.00	0.00
v_{18}	-0.08	0.03	0.27	0.17	0.26	0.13
v_{19}	-13.32	11.69	0.00	0.00	0.00	0.00
v_{20}	-0.19	0.59	0.29	0.00	0.11	0.00

classifier (blue and red ROC curves of figure 33).

Figure 36 compares the ROC curves generated by the trained λ_{CHMM}^{CPR}, $\lambda_{CHMM}^{\sim CPR}$, and λ_{CHMM}^{CPR} - $\lambda_{CHMM}^{\sim CPR}$ models (in solid lines) and the corresponding initial models before training (in dashed lines). We notice that the BW-training improves significantly the classification performance for the non-CPR model (blue ROC curves of figure 36). However, the BW-trained CPR model slightly underperfoms compared to the non-trained CPR model in terms of the area under ROC measure. Nonetheless, the BW-trained baseline CHMM classifier that combines the CPR and non-CPR models has the largest area under ROC curve.

4.3.3.2 Baseline continuous HMM results

Tables 11 and 12 display the initial and the Baum-Welch trained λ_{CHMM}^{CPR} and $\lambda_{CHMM}^{\sim CPR}$ models parameters, respectively. In these tables, we notice that, for both λ_{CHMM}^{CPR} and $\lambda_{CHMM}^{\sim CPR}$, the BW-training updates resulted in one dominant component of the Gaussian mixture (e.g. component 3 of state 1 and component 2 of state 2 of λ_{CHMM}^{CPR}). For both models, the dominant component in

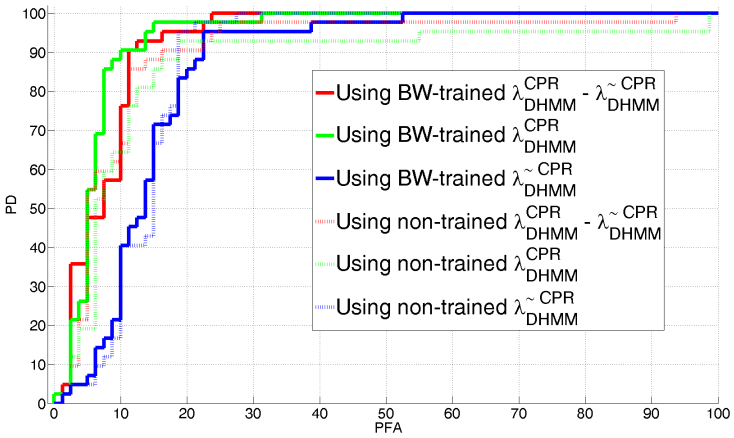


Figure 33. ROCs generated by the BW-trained (solid lines) and initial (dashed lines) DHMM models

each state is the one with the lowest magnitude $|\bar{\mathbf{u}}|$. However, for λ_{CHMM}^{CPR}, the dominant component is the one that has the highest covariance along $\bar{\mathbf{u}}^y$, while, for $\lambda_{CHMM}^{\sim CPR}$, the dominant component is the one that has the lowest covariance along $\bar{\mathbf{u}}^y$. Converging to a single dominant component, specially for $\lambda_{CHMM}^{\sim CPR}$ where state 1 component 3 weight is 0.92 (refer to table12), suggests that $\lambda_{CHMM}^{\sim CPR}$ state 1 Gaussian mixture could have been modeled by a lesser number of components.

TABLE 11

Baseline CHMM model parameters of λ_{CHMM}^{CPR}

Initial Model

A

	s_1	s_2
s_1	0.5	0.5
s_2	0.5	0.5

Priors

s_1	1.00
s_2	0.00

B Means			
		$\bar{u}^x$	$\bar{u}^y$
s_1	c_{11}	-0.16	1.09
	c_{12}	0.57	1.11
	c_{13}	0.04	0.16
s_2	c_{21}	-0.32	-0.76
	c_{22}	0.01	-0.21
	c_{23}	2.33	-0.56

Covariance		
	$\bar{u}^x$	$\bar{u}^y$
cov_{11}	0.10	0.13
cov_{12}	0.20	0.21
cov_{13}	0.14	0.10
cov_{21}	0.16	0.10
cov_{22}	0.10	0.10
cov_{23}	0.33	0.20

Coefficients	
	w
g_{11}	0.33
g_{12}	0.33
g_{13}	0.33
g_{21}	0.33
g_{22}	0.33
g_{23}	0.33

Trained Model

A

	s_1	s_2
s_1	0.66	0.34
s_2	0.23	0.77

Priors

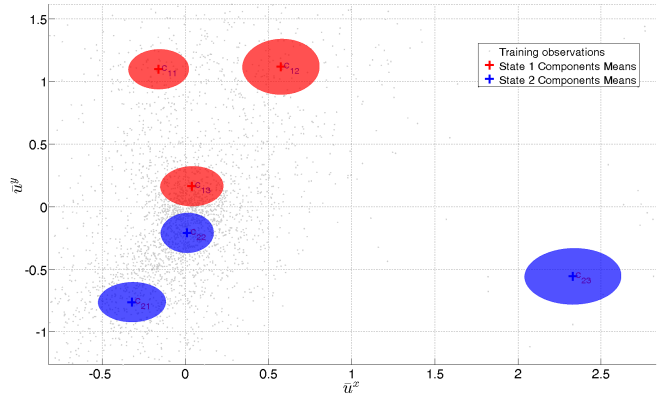
s_1	1.00
s_2	0.00

B Means			
		$\bar{u}^x$	$\bar{u}^y$
s_1	c_{11}	0.23	1.09
	c_{12}	0.21	0.93
	c_{13}	0.09	0.31
s_2	c_{21}	-0.20	-0.67
	c_{22}	-0.07	-0.35
	c_{23}	-0.10	-0.42

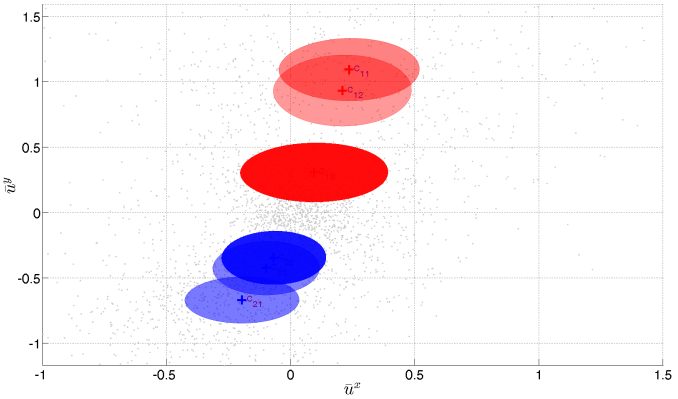
Covariance		
	$\bar{u}^x$	$\bar{u}^y$
cov_{11}	0.32	0.23
cov_{12}	0.29	0.31
cov_{13}	0.20	0.35
cov_{21}	0.21	0.13
cov_{22}	0.18	0.17
cov_{23}	0.18	0.17

Coefficients	
	w
g_{11}	0.18
g_{12}	0.13
g_{13}	0.69
g_{21}	0.21
g_{22}	0.60
g_{23}	0.19

Figure 34 displays the observation vectors used to train the CPR model (first cross validation set). In this figure, we show the λ_{CHMM}^{CPR} model parameters for both the initial model (a) and the BW-



(a)



(b)

Figure 34. Gaussian mixtures of (a) the initial λ^{CPR}_{CHMM} model, and (b) the BW-trained λ^{CPR}_{CHMM} model

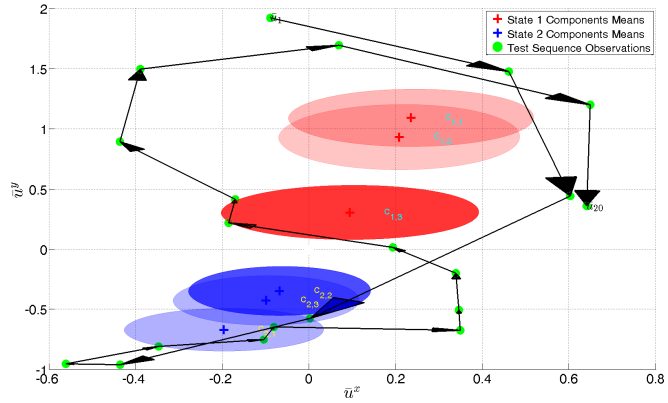


Figure 35. Illustration of testing the CPR sequence of figure 32(a) with λ^{CPR}_{CHMM}

TABLE 12

Baseline CHMM model parameters of $\lambda_{CHMM}^{\sim CPR}$

Initial Model			**B Means**				**Covariance**			**Coefficients**	
A					$\bar{u}^x$	$\bar{u}^y$		$\bar{u}^x$	$\bar{u}^y$		w
	s_1	s_2		c_{11}	-0.26	9.70	cov_{11}	59.96	1.43	g_{11}	0.33
s_1	0.5	0.5	s_1	c_{12}	-1.04	0.55	cov_{12}	0.72	0.62	g_{12}	0.33
s_2	0.5	0.5		c_{13}	0.02	0.06	cov_{13}	0.10	0.10	g_{13}	0.33
Priors				c_{21}	0.42	-0.25	cov_{21}	0.13	0.14	g_{21}	0.33
s_1	0.50		s_2	c_{22}	-0.49	-0.65	cov_{22}	0.66	0.62	g_{22}	0.33
s_2	0.50			c_{23}	-0.01	-0.05	cov_{23}	0.10	0.10	g_{23}	0.33

Trained Model			**B Means**				**Covariance**			**Coefficients**	
A					$\bar{u}^x$	$\bar{u}^y$		$\bar{u}^x$	$\bar{u}^y$		w
	s_1	s_2		c_{11}	-3.95	7.92	cov_{11}	25.97	9.56	g_{11}	0.01
s_1	0.40	0.60	s_1	c_{12}	-0.25	1.07	cov_{12}	1.05	0.81	g_{12}	0.08
s_2	0.25	0.75		c_{13}	-0.01	0.05	cov_{13}	0.17	0.10	g_{13}	0.92
Priors				c_{21}	0.00	-0.02	cov_{21}	0.14	0.10	g_{21}	0.36
s_1	0.50		s_2	c_{22}	-0.44	-0.97	cov_{22}	1.50	1.79	g_{22}	0.03
s_2	0.50			c_{23}	0.00	0.00	cov_{23}	0.14	0.10	g_{23}	0.61

trained model (b). In particular, we display the means of the states components where $c_{i,j}$ denotes the center of component j of state i, for $n \in \{1, 2\}$ and $m \in \{1, 2, 3\}$. The corresponding covariance is represented by an ellipse with dimensions proportional to the covariance in each direction $\bar{\mathbf{u}}^x$ and $\bar{\mathbf{u}}^y$; state 1 Gaussian components are plotted in red while state 2 in blue; and color opacity of each component is proportional to its coefficient in the Gaussian mixture. In figure 34(b), we notice that the updated means shifted overall towards the center of the ($\bar{\mathbf{u}}^x$;$\bar{\mathbf{u}}^y$) plane in such a way that the dominant component has the lowest magnitude $|\bar{\mathbf{u}}^y|$.

Figure 35 illustrates the testing process of the CPR sequence of figure 32(a) by λ_{CHMM}^{CPR}. Similarly to the DHMM model, the first and last observation (upward motion) are closer to the means of the Gaussian mixture of state 1 and thus have higher probabilities in state 1. Similarly, the middle observations (downward motion) have higher probabilities in the Gaussian mixture of state 2. The optimal state sequence, assigned by the Viterbi algorithm, to this test sample is **11122222222221111111**.

Figure 36 compares the ROC curves generated by the trained λ_{CHMM}^{CPR}, $\lambda_{CHMM}^{\sim CPR}$, and λ_{CHMM}^{CPR} - $\lambda_{CHMM}^{\sim CPR}$ models (in solid lines) and the corresponding initial models before training (in dashed lines). We notice that the BW-training improves significantly the classification performance for the CPR model (green ROC curves of figure 36). However, the BW-trained non-CPR model underperfoms compared to the non-trained non-CPR model in terms of the area under ROC measure.

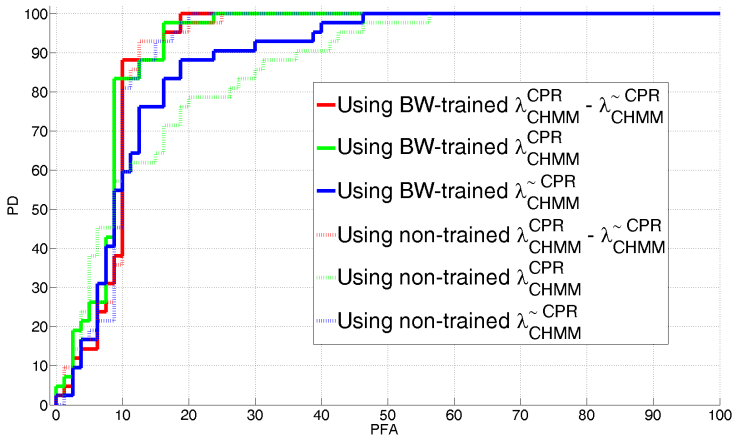


Figure 36. ROCs generated by the BW-trained (solid lines) and initial (dashed lines) CHMM models.

Overall, the BW-training improves the performance of baseline CHMM classifier that combines the CPR and non-CPR models.

4.3.4 Video shot classification using ensemble HMM

4.3.4.1 Fitting HMM models to individual sequences

For each sequence $O_r \in \mathfrak{D}_{Trn}$, we build an HMM model, λ_r,based on the observations $\{\bar{\mathbf{u}}_r^1, \cdots, \bar{\mathbf{u}}_r^T\}$. We assume that each λ^r has two states $s_1^{(r)}$ and $s_2^{(r)}$: one represents the upward motion of the hands and the second represents the downward motion. The states means are estimated using:

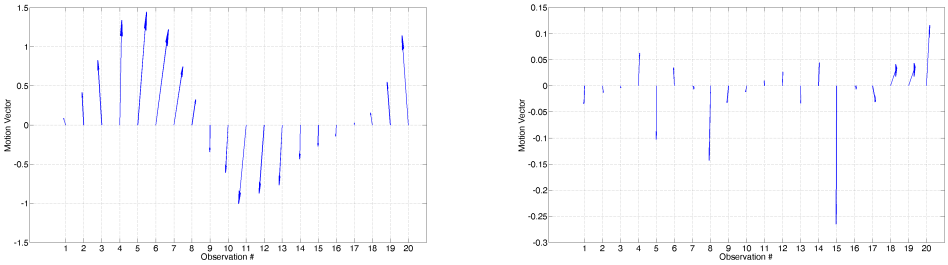
$$\begin{cases} s_1^{(r)} = average\{\bar{\mathbf{u}}_r \in O_r | \bar{u}_r^y \geq 0\} \\ s_2^{(r)} = average\{\bar{\mathbf{u}}_r \in O_r | \bar{u}_r^y < 0\}. \end{cases} \tag{104}$$

The remaining parameters $(\mathbf{A}, \mathbf{B}, \pi)$ of λ^r are initialized as follows. The priors, π_1 and π_2, are set to $[1; 0]$ for CPR sequences and $[0.5; 0.5]$ for non CPR sequences. The transition matrix is initialized using:

$$A = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}.$$

The initialization of the emission probabilities for each state depends whether we use a discrete or a continuous HMM model.

- For the discrete case, the emission matrix, B, is initialized based on the states and a codebook of size M=20. In the case of individual non-CPR sequences models, the codewords $\{\mathbf{v}_m, m = 1, \cdots, M\}$ are the actual sequence observations $\{\bar{\mathbf{u}}_r^t, t = 1, \cdots, T\}$ of O_r. For the



(a) CPR Sequence O_1 (b) non-CPR Sequence O_2

Figure 37. Motion vectors of two sample sequences

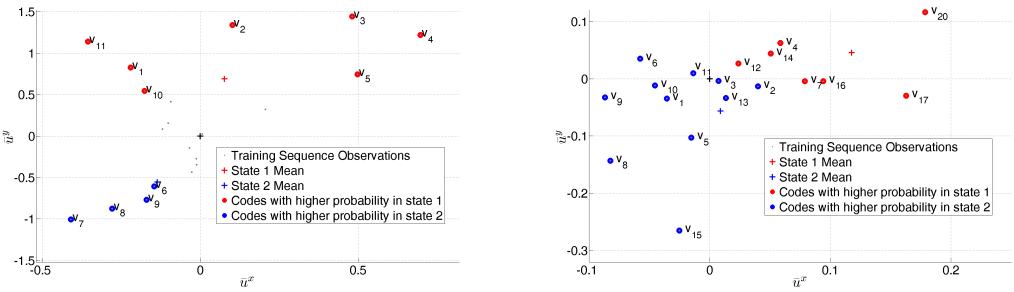
CPR sequences models, we discard codewords that have low magnitude ($|\bar{\mathbf{u}}^y| < 0.5$). This will prevent observations with small motion vectors (mainly from non-CPR sequences) from having high probabilities in those codewords, i.e. in individual CPR sequences models. Finally, the emission probabilities are initialized based on the distance between the codewords and the states means, using (103).

- For the continuous case, the emission probabilities are modeled by one Gaussian component for each state. The choice of a single-component Gaussian mixture is motivated by the small number of observations and the uni-modality of the CPR sequences observations. This assumption is also valid for non-CPR sequences that can be modeled by a Gaussian distribution with a large variance. Therefore, for each state, the parameters of the Gaussian distribution are the mean of the state, computed using (104), and the statistical covariance obtained from the respective observations. In particular, we use:

$$\begin{cases} cov_r^{(1)} = Cov\{\bar{\mathbf{u}}_r = (\bar{u}_r^x, \bar{u}_r^y) \in O_r | \bar{u}_r^y \geq 0\} \\ cov_r^{(2)} = Cov\{\bar{\mathbf{u}}_r = (\bar{u}_r^x, \bar{u}_r^y) \in O_r | \bar{u}_r^y < 0\}. \end{cases}$$

After initialization, the transition probabilities and the emission probability parameters of each model are fine-tuned using either the discrete or continuous version of the Baum-Welch algorithm [9] and the corresponding training observation. As mentioned earlier in section 3.3.1.1, the use of only one observation sequence to form and train an HMM leads to over-fitting, and subsequently to the desired property that the likelihood of each sequence with respect to its model being higher than the other sequences likelihoods in the same model. Let $\{\lambda_r\}_{r=1}^{R_n}$ be the set of trained individual models for the n^{th} crossvalidation fold.

Let $O_1 = \{\bar{\mathbf{u}}_1^1, \cdots, \bar{\mathbf{u}}_1^T\}$ and $O_2 = \{\bar{\mathbf{u}}_2^1, \cdots, \bar{\mathbf{u}}_2^T\}$ be, respectively, a CPR sequence and a non-CPR sequence from the subset $\mathfrak{D}_{Trn}$ of the training data. In figures 37(a) and (b), we show



(a) (b)

Figure 38. Training observations, states means, and codewords of the models :(a) λ_1^{DHMM}, and (b) λ_2^{DHMM} of the sequences O_1 and O_2 of figures 37(a)-(b)

the sequence of 20 motion features vectors of O_1 and O_2, respectively. Let λ_i^{DHMM} and λ_i^{CHMM} denote, respectively, the BW-trained DHMM and CHMM models for sequence O_i, $i \in \{1, 2\}$.

Fitting discrete HMM models to individual sequences: Tables 13 and 14 show the DHMM models parameters for sequences O_1 and O_2, respectively. For each model, the initial and the Baum-Welch trained parameters (namely the state means, the state transition probabilities, the codewords, and their emission probabilities in each state) are shown. Since the motion feature vectors are 2-dimensional, we visualize the parameters of λ_1^{DHMM} and λ_2^{DHMM} in the $(\bar{u}_r^x; \bar{u}_r^y)$ plane in figures 38(a) and (b) respectively. From tables 13 and 14 and the respective figures 38(a) and 38(b), we notice that:

TABLE 13

λ_1^{DHMM} model parameters

Means			**Initial A**		**A**	
	$\bar{u}^x$	$\bar{u}^y$	s_1	s_2	s_1	s_2
s_1	0.08	0.69	0.5	0.5	0.91	0.09
s_2	-0.14	-0.56	0.5	0.5	0.13	0.87
Codes			**Initial B**		**B**	
v_1	-0.22	0.83	0.40	0.00	0.08	0.00
v_2	0.10	1.34	0.00	0.00	0.08	0.00
v_3	0.48	1.44	0.00	0.00	0.08	0.00
v_4	0.70	1.22	0.00	0.00	0.08	0.00
v_5	0.50	0.75	0.00	0.00	0.08	0.00
v_6	-0.15	-0.61	0.00	0.47	0.00	0.62
v_7	-0.41	-1.01	0.00	0.00	0.00	0.12
v_8	-0.28	-0.87	0.00	0.12	0.00	0.12
v_9	-0.17	-0.77	0.00	0.42	0.00	0.12
v_{10}	-0.18	0.55	0.60	0.00	0.50	0.00
v_{11}	-0.36	1.14	0.00	0.00	0.08	0.00

TABLE 14

λ_2^{DHMM} model parameters

Means			**Initial A**		**A**	
	$\bar{u}^x$	$\bar{u}^y$	s_1	s_2	s_1	s_2
s_1	0.12	0.05	0.5	0.5	0.53	0.47
s_2	0.01	-0.06	0.5	0.5	0.46	0.54
Codes			**Initial B**		**B**	
v_1	-0.04	-0.03	0.01	0.08	0.01	0.08
v_2	0.04	-0.01	0.03	0.07	0.03	0.07
v_3	0.01	0.00	0.02	0.08	0.02	0.07
v_4	0.06	0.06	0.11	0.02	0.09	0.01
v_5	-0.02	-0.10	0.01	0.08	0.01	0.09
v_6	-0.06	0.03	0.04	0.06	0.04	0.06
v_7	0.08	0.00	0.08	0.03	0.08	0.03
v_8	-0.08	-0.14	0.01	0.07	0.02	0.08
v_9	-0.09	-0.03	0.02	0.07	0.02	0.07
v_{10}	-0.05	-0.01	0.02	0.08	0.02	0.08
v_{11}	-0.01	0.01	0.03	0.07	0.03	0.07
v_{12}	0.02	0.03	0.06	0.05	0.06	0.04
v_{13}	0.01	-0.03	0.00	0.09	0.00	0.09
v_{14}	0.05	0.04	0.09	0.02	0.08	0.02
v_{15}	-0.03	-0.27	0.01	0.06	0.02	0.08
v_{16}	0.09	0.00	0.10	0.02	0.09	0.02
v_{17}	0.16	-0.03	0.10	0.02	0.09	0.02
v_{18}	0.34	0.03	0.08	0.01	0.10	0.01
v_{19}	0.36	0.04	0.07	0.00	0.10	0.01
v_{20}	0.18	0.12	0.11	0.01	0.10	0.01

1. The means and the codewords of λ_1^{DHMM} have higher magnitude motion vectors compared to the non-CPR model λ_2^{DHMM}.

2. The transition matrix A of λ_1^{DHMM} has a higher diagonal entries than that of λ_2^{DHMM}. Precisely, in λ_1^{DHMM}, the entry $a_{11} = 0.91$ is exactly the ratio of (1) the number of state 1 self-transitions (nine state 1 to state 1 transitions as it can be seen in figure 37(a)) to (2) the total number of transitions from state 1 (eleven transition occurrences). Similarly for the entry a_{22}, that ratio is 7 to 8 as it can be deduced from the same figure 37(a). However, for λ_2^{DHMM}, the transitions from state 1 to either state 1 or state 2 (and vice versa) are almost equiprobable.

3. For the emission probabilities, we notice that, for the BW-trained λ_1^{DHMM}, one codeword (v_{10} in state 1 and v_6 in state 2) has the highest emission probability while few other codewords ($\{v_1, \cdots, v_5, v_{11}\}$ in state 1 and $\{v_7, \cdots, v_9\}$ in state 2) have the same relatively low probability. In both states, the codeword with highest probability is the one closest to the state mean (refer

to table 13 and to the subsequent illustration of the λ_1^{DHMM} parameters in the $(\bar{u}_r^x; \bar{u}_r^y)$ plane in figure 38(a)). For the BW-trained non-CPR model λ_2^{DHMM}, all the codewords have relatively low magnitude motion vector and their respective emission probabilities in each state are low and comparable (≤ 0.10 as can be seen in table 14). In conclusion, the emission probability density function of each state of λ_1^{DHMM} can be approximated by a discrete normal distribution centered around the mean of that state. Similarly, the emission probability density function of each state of λ_2^{DHMM} can be approximated by a discrete uniform distribution. These findings will be discussed in the following section for the continuous case.

Fitting <u>continuous</u> HMM models to individual sequences: Similarly to the discrete case, we use the same sequences O_1 and O_2 (shown in figure 37) to fit two CHMM models: λ_1^{CHMM} and λ_2^{CHMM}. The initial and BW-updated parameters of λ_1^{CHMM} and λ_2^{CHMM} are shown in tables 15 and 16, respectively.

TABLE 15

λ_1^{CHMM} model parameters

Initial Model

A

	s_1	s_2
s_1	0.5	0.5
s_2	0.5	0.5

B	**Means**			**Covariance**			**Coefficients**	
		$\bar{u}^x$	$\bar{u}^y$		$\bar{u}^x$	$\bar{u}^y$		w
s_1	c_{11}	0.08	0.69	cov_{11}	0.108	0.256	g_{11}	1.0
s_2	c_{21}	-0.14	-0.56	cov_{21}	0.021	0.094	g_{21}	1.0

Trained Model

A

	s_1	s_2
s_1	0.90	0.10
s_2	0.12	0.88

B	**Means**			**Covariance**			**Coefficients**	
		$\bar{u}^x$	$\bar{u}^y$		$\bar{u}^x$	$\bar{u}^y$		w
s_1	c_{11}	0.08	0.73	cov_{11}	0.105	0.228	g_{11}	1.0
s_2	c_{21}	-0.12	-0.50	cov_{21}	0.019	0.112	g_{21}	1.0

We notice that the BW-training does not change the CHMM model parameters significantly except for the transition matrices. For the covariance values, it is worth noting that variance along $\bar{u}_r^y$ is higher than the variance along $\bar{u}_r^x$ direction.

4.3.4.2 Similarity matrix computation

In the first step of the eHMM, we built a set $\{\lambda_r\}_{r=1}^{R_n}$ of trained individual HMM models for each crossvalidation fold. In this step, we test each observation sequence O_j in each model λ_i, for $1 \leq i, j \leq R_n$. This test produces (1) a log-likelihood value $\mathbf{L}_{ij} = log(Pr(O_j|\lambda_i))$ that is the probability of sequence O_j being generated by λ_i and (2) a path p_{ij} that represents the optimal

TABLE 16

λ_2^{CHMM} model parameters

Initial Model

A

	s_1	s_2
s_1	0.5	0.5
s_2	0.5	0.5

B	**Means**			**Covariance**			**Coefficients**	
		$\bar{u}^x$	$\bar{u}^y$		$\bar{u}^x$	$\bar{u}^y$		w
s_1	c_{11}	0.12	0.05	cov_{11}	0.025	0.010	g_{11}	1.0
s_2	c_{21}	0.01	-0.06	cov_{21}	0.010	0.010	g_{21}	1.0

Trained Model

A

	s_1	s_2
s_1	0.28	0.72
s_2	0.30	0.70

B	**Means**			**Covariance**			**Coefficients**	
		$\bar{u}^x$	$\bar{u}^y$		$\bar{u}^x$	$\bar{u}^y$		w
s_1	c_{11}	0.08	0.02	cov_{11}	0.016	0.010	g_{11}	1.0
s_2	c_{21}	0.04	-0.03	cov_{21}	0.014	0.010	g_{21}	1.0

(most likely) sequence of states of λ_i that would have generated O_j. This path is computed using the Viterbi algorithm [8]. We define the path mismatch penalty, denoted $\mathbf{P}_{ij}$, as the edit distance [51] between the optimal paths p_{ij} and p_{ii}, that resulted from testing O_j in λ_i and O_i in λ_i respectively. Given the log-likelihood and the path mismatch penalty matrices, we define the similarity matrix, $\mathbf{S}$, as a weighted sum of $\mathbf{L}$ and $\mathbf{P}$ using a mixing factor $\alpha \in [0, 1]$ (refer to equation (74) in section 3.3.1.3).

Similarity matrix computation using discrete HMMs: To illustrate the similarity matrix computation using discrete HMMs, we reconsider the sample sequences O_1 and O_2, shown in figure 37, and their respective DHMM models λ_1^{DHMM} and λ_2^{DHMM}, displayed in tables 13 and 14. In figure 39, we illustrate the testing of the sequences O_1 and O_2 in λ_1^{DHMM} and λ_2^{DHMM}.

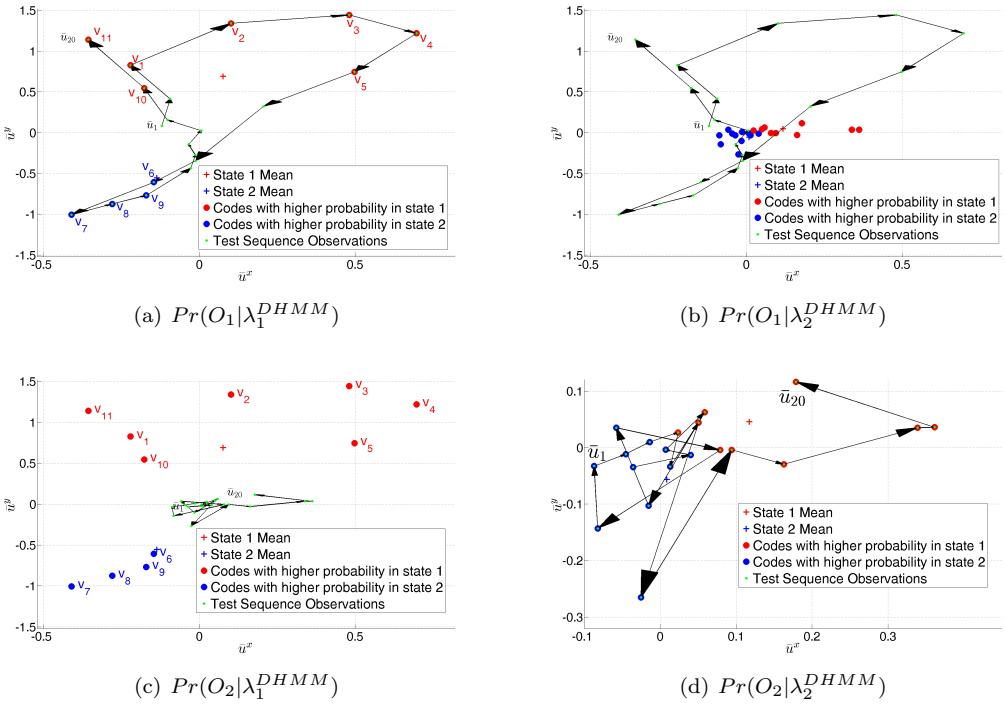
The log-likelihoods and the consequent optimal Viterbi paths obtained by testing O_1 and O_2 in λ_1^{DHMM} and λ_2^{DHMM} are summarized in tables 17 and 18.

TABLE 17

Log-likelihoods of the two sequences in figure 37 in the models generated by these sequences

$\mathbf{L}_{ij}$	λ_1^{DHMM}	λ_2^{DHMM}
O_1	-9.27	-21.50
O_2	-38	-3.21

From the paths in table 18, we use the "*edit*" distance measure to deduce the path mismatch penalty terms (shown in table 19) for sequences O_1 with regards to O_2 and vice versa. Figures 40(a) and 40(b) show the log-likelihood and path mismatch penalty matrices for all the training sequences $\mathfrak{D}_{Trn}$. Figures 40(c) and 40(d) show the resulting similarity matrix using $\alpha = .5$ and



(a) $Pr(O_1|\lambda_1^{DHMM})$ (b) $Pr(O_1|\lambda_2^{DHMM})$

(c) $Pr(O_2|\lambda_1^{DHMM})$ (d) $Pr(O_2|\lambda_2^{DHMM})$

Figure 39. Illustration of the testing of sequences O_1 and O_2 in λ_1^{DHMM} and λ_2^{DHMM}

TABLE 18

Viterbi paths resulting from testing O_1 and O_2 in models λ_1^{DHMM} and λ_2^{DHMM}. p_{ij} refers to the optimal path obtained by testing sequence i with model j.

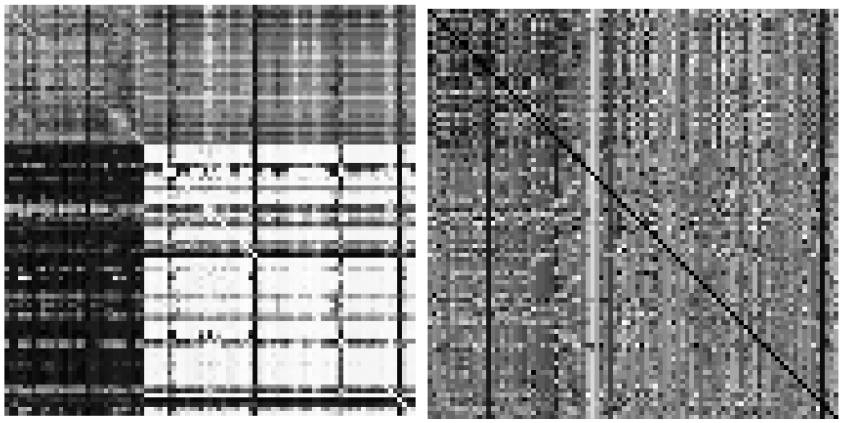
	$\bar{\mathbf{u}}_1$	$\cdots$																		$\bar{\mathbf{u}}_{20}$
p_{11}	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1
p_{21}	1	1	1	1	2	1	1	2	2	1	1	1	2	1	2	1	2	1	1	1
p_{12}	2	2	2	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2
p_{22}	2	2	2	1	2	2	1	2	2	2	2	2	2	1	2	1	1	1	1	1

$\alpha = .1$ respectively. In these matrices, the indices are rearranged so that the first entries correspond to the CPR sequences $\mathfrak{D}_{Trn}^{CPR}$ and the latter ones correspond to the non-CPR sequences $\mathfrak{D}_{Trn}^{\sim CPR}$. In these figures, dark pixels correspond to small values of the log-likelihood and path mismatch penalty and bright pixels correspond to larger entries of the corresponding matrices. Note that in the case of the log-likelihood matrix of figure 40(a), the diagonal blocks are brighter than the off-diagonal blocks. Conversely, in the path mismatch penalty matrix of figure 40(b)), the diagonal blocks are darker than the off-diagonal blocks. Thus, both the log-likelihood and the path mismatch penalty provide complementary measures for the similarity between signatures. This was the motivation for combining both matrices into the similarity $\mathbf{S}$ using the mixing factor $\alpha = 0.5$ in figure 40(c) and

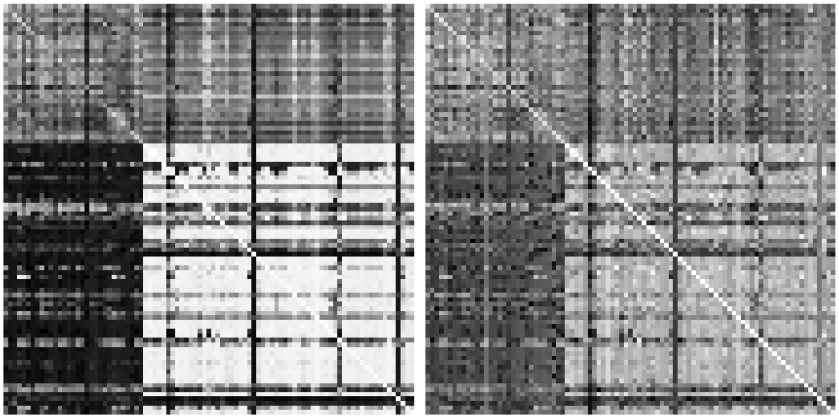
TABLE 19

Path mismatch penalty of the two sequences in figure 37 in the models generated by these sequences

$\mathbf{P}_{ij}$	λ_1^{DHMM}	λ_2^{DHMM}
O_1	0	9
O_2	8	0



(a) Log-likelihood $\mathbf{L}$ (b) Penalty $\mathbf{P}$



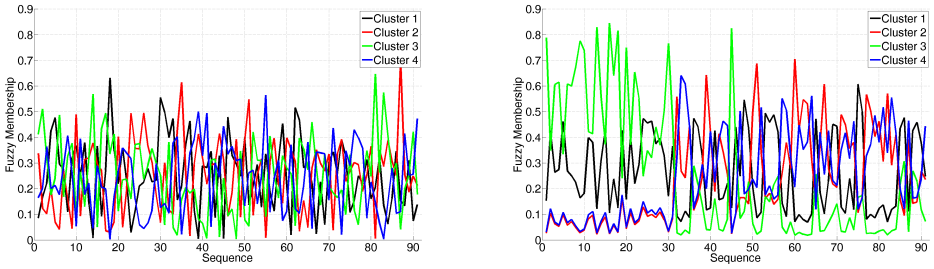
(c) Similarity $\mathbf{S}$ with $\alpha = .5$ (d) Similarity $\mathbf{S}$ with $\alpha = .1$

Figure 40. Log-likelihood, path mismatch penalty, and similarity matrices for the sequences in $\mathfrak{D}_{Trn}$

$\alpha = 0.1$ in figure 40 (d). In the next step of the eHMM, we use the similarity matrix of figure 40(c) to partition $\mathfrak{D}_{Trn}$ into groups of similar sequences.

4.3.4.3 Clustering results

The similarity matrix $\mathbf{S}$ of figure 40(c) is transformed to a symmetric distance matrix $\mathbf{D}$ using (77). Any relational-based clustering algorithm can be applied, using $\mathbf{D}$, to partition the training data subset $\mathfrak{D}_{Trn}$ into K clusters. In our experiments in this chapter, we use the relational



(a) (b)

Figure 41. Fuzzy membership of the training sequences in each cluster: (a) initial membership (b) after FLeCK convergence

clustering with learnable cluster dependent kernels (FLeCK) clustering algorithm [52] with $K = 4$.

Figure 41 shows the fuzzy membership of the training sequences in each cluster. Initially, a random fuzzy partition matrix $U^{(0)}$ of size $(R_n \times K)$ was created. Those initial membership values are shown in figure 41(a). The FLeCK algorithm converges when the membership values do not change significantly from one iteration to the next ($\|(U^{(t)} - U^{(t-1)})\| < 10^{-6}$). In our experiment, FLeCK converged after 118 iterations. The resulting fuzzy memberships are shown in figure 41(b). The variance of each cluster is displayed in table 20. As shown in figure 41(b), the

TABLE 20

Variance in each cluster after FLeCK convergence

Cluster	σ_1	σ_2	σ_3	σ_4
Variance	0.13	0.11	0.13	0.11

first sequences of $\mathfrak{D}_{Trn}$ have high fuzzy membership in clusters 1 and 3, while the latter ones have higher memberships in clusters 2 and 4. Recall that the indices of $\mathfrak{D}_{Trn}$ are rearranged so that the first elements correspond to the CPR sequences $\mathfrak{D}_{Trn}^{CPR}$ and the latter ones correspond to the non-CPR sequences $\mathfrak{D}_{Trn}^{\sim CPR}$, i.e. $\mathfrak{D}_{Trn} = \{\mathfrak{D}_{Trn}^{CPR}, \mathfrak{D}_{Trn}^{\sim CPR}\}$. Thus, clusters 1 and 3 are composed mainly of CPR sequences and few non-CPR sequences and clusters 2 and 4 comprise only non-CPR sequences. Figure 42 shows the partition of the training data in terms of the number of CPR and non-CPR sequences in each cluster. In figure 43, we visualize the actual sequences belonging to each cluster (CPR sequences are displayed in red while non-CPR sequences are shown in blue). In particular, for each sequence, we plot only the $\bar{u}^y$ component of each observation. We notice that all but one of the CPR sequences belonging to cluster 1 start with low magnitude $\bar{u}^y$ up to the 5^{th} observation. Most of these sequences have a high positive $\bar{u}^y$ value at observations $\{6, \cdots, 10\}$,

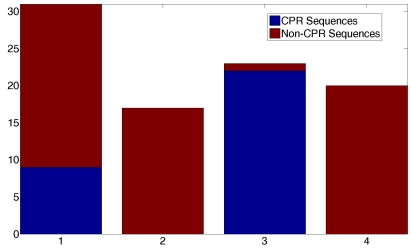


Figure 42. Distribution of the sequences in each cluster per class (CPR vs. non-CPR)

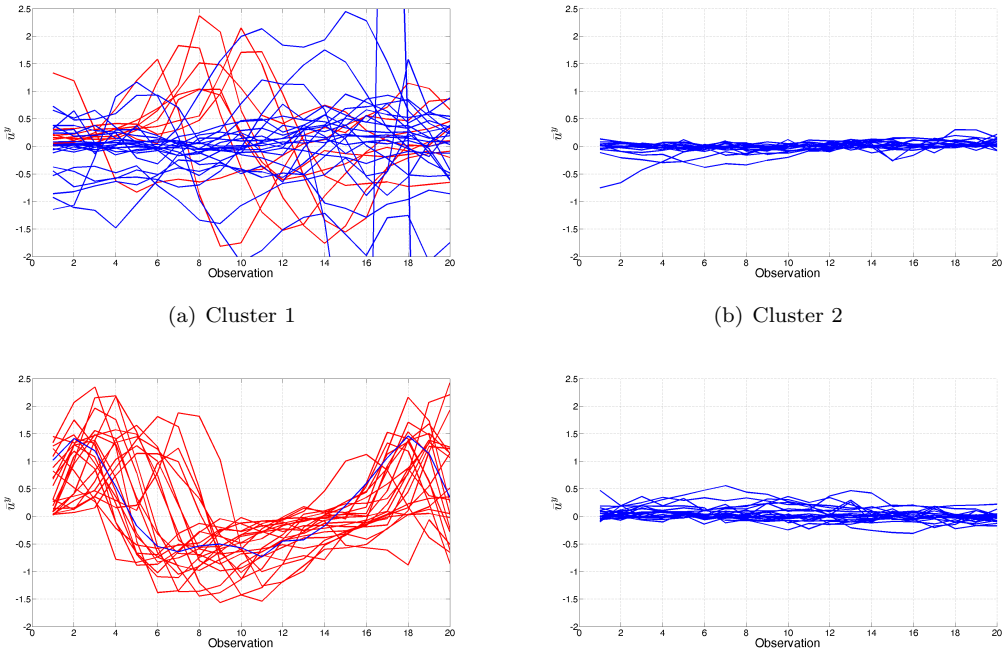
negative $\bar{u}^y$ values around observations $\{12, \cdots, 16\}$, and then back to low magnitude $\bar{u}^y$ values at the last observations. Cluster 3 comprises 22 CPR sequences and 1 non-CPR sequence. For all these sequences, including the non-CPR one, the first and last observations have high $\bar{u}^y$ values (upward motion) while the middle observations have negative $\bar{u}^y$'s (downward motion). Finally, Clusters 2 and 4 are composed mainly of low-magnitude $\bar{u}^y$ value sequences with random number of transition between negative and positive $\bar{u}^y$'s.

The above figures illustrate the purpose of step 2 of the proposed eHMM approach, i.e. grouping similar signatures into clusters. To summarize, using the distance matrix $\mathbf{D}$ of (77), the R sequences are clustered into K clusters. The objective of this clustering step is to group similar observations in the log-likelihood space. The observations forming each cluster are then used to learn multiple HMM models.

4.3.4.4 Clusters models initialization and training

For each cluster in figure 42, one or two HMM models are built based on the elements belonging to that cluster: one for CPR sequences and/or one for non-CPR sequences. In particular, the initialization of the priors and transition matrices for each model is performed by averaging the corresponding parameters of the individual models of the sequences belonging to that cluster. We assume that each model has $N = 2$ states. The initialization of the emission probabilities for the discrete case is straightforward. That is, the states' means, s_n, of the cluster model are obtained by averaging the states means of the individual models. Finally, the codes v_m, $m = 1, \cdots, M$, of the cluster model are obtained by clustering the codes of the individual models into $M = 20$ clusters.

For the continuous case, we further assume that each state has three Gaussian components. For each cluster model, the means and covariances of state n components are obtained by clustering the state n observations of the sequences belonging to the cluster into three clusters using k-means algorithm [23]. Precisely, the mean of each component is the center of one of the resulting clusters



(a) Cluster 1

(b) Cluster 2

(c) Cluster 3

(d) Cluster 4

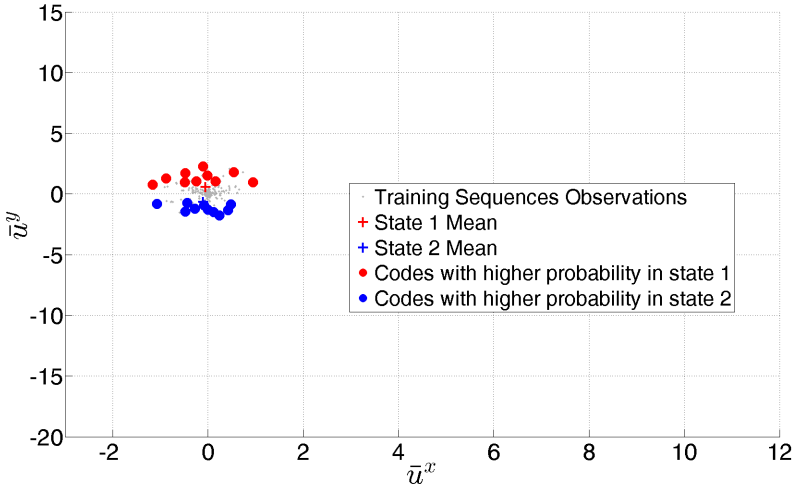
Figure 43. Sequences belonging to each cluster, only $\bar{u}^y$ motion vector component is plotted for each sequence' observation.

and the covariance is estimated using the observations belonging to that same cluster.

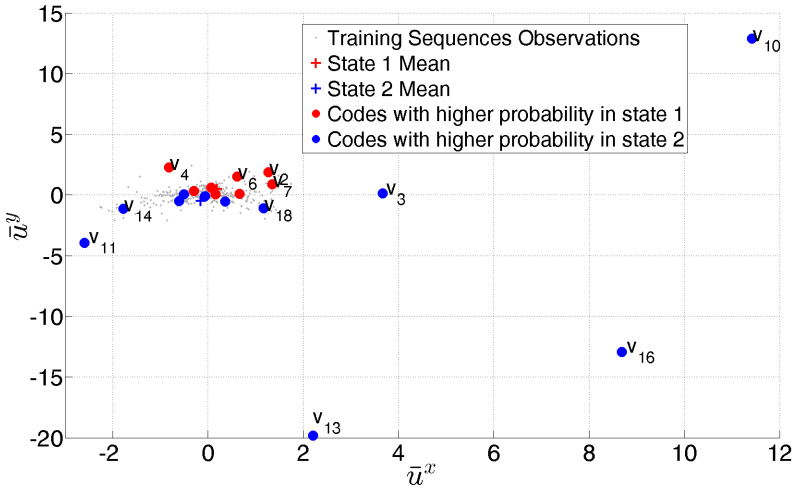
After initialization, and depending on the cluster size and homogeneity, one of the training schemes described earlier, is devised. Let $\lambda_k^{CPR,DHMM}$ and $\lambda_k^{\sim CPR,DHMM}$ denote the trained DHMM models for the CPR and non-CPR sequences belonging to cluster k.

Clusters DHMM models initialization and training: In this case, we simply use the notation λ_k^{CPR} and $\lambda_k^{\sim CPR}$ to refer to the CPR and non-CPR cluster k models. Figure 44(a) shows the CPR training sequences of cluster 1 and the parameters of the model, λ_1^{CPR}, built using those sequences. Similarly, in figure 44(b), we show the non-CPR sequences observations of cluster 1 and their respective $\lambda_1^{\sim CPR}$ parameters. We notice that the codes of λ_1^{CPR} have relatively high magnitude motion vectors. The codes closest to the mean of each state have higher B values in that state. However, for the $\lambda_1^{\sim CPR}$ model, the majority of the codes have relatively low magnitude motion vectors. A few codes of $\lambda_1^{\sim CPR}$ have very high magnitude motion vectors but their emission probability is very low. This suggests that those codes represent observations from outlier non-CPR sequence(s) in the training data.

Cluster 2 is composed uniquely of non-CPR sequences. Thus, we set its corresponding λ_2^{CPR}



(a) λ_1^{CPR}



(b) $\lambda_1^{\sim CPR}$

Figure 44. Training sequences and λ_1^{CPR} DHMM model parameters of cluster 1

to $\emptyset$ and, by definition, returns a default likelihood value $minProb = -40$. In figure 45, we show the non-CPR sequences belonging to cluster 2 and the $\lambda_1^{\sim CPR}$ model parameters. As expected, $\lambda_1^{\sim CPR}$ have relatively low magnitude codes and states means, with codes closest to the mean of each state have higher B values in that state.

Cluster 3 is composed of mainly CPR sequences with only 1 non-CPR sequence. In this case, the non-CPR cluster 3 model, $\lambda_3^{\sim CPR}$, is defined as the individual model of that sequence. The CPR model, λ_3^{CPR} and the CPR sequences of cluster 3 are shown in figure 46.

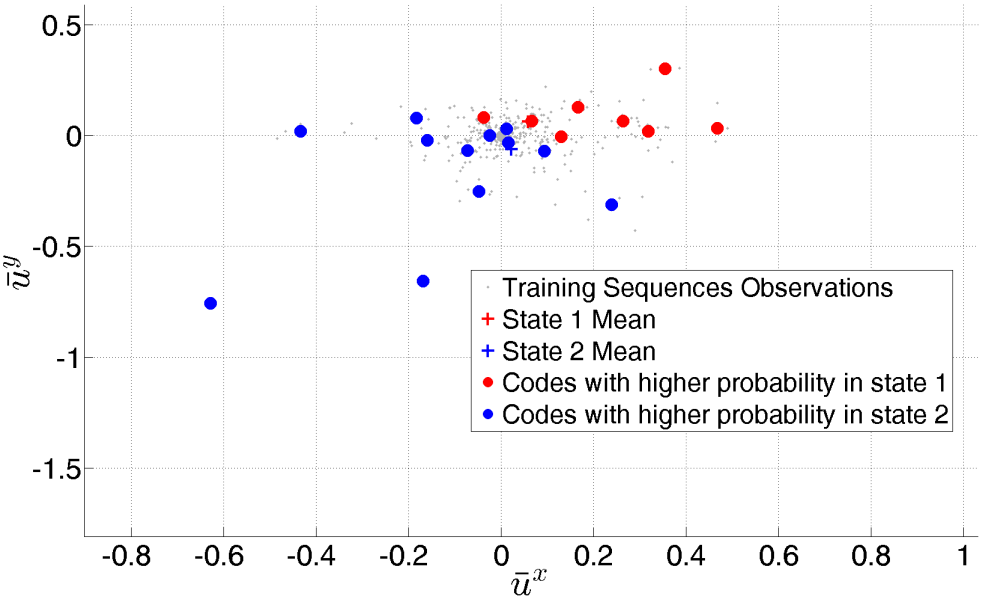


Figure 45. Training sequences and $\lambda_2^{\sim CPR}$ DHMM model parameters of cluster 2

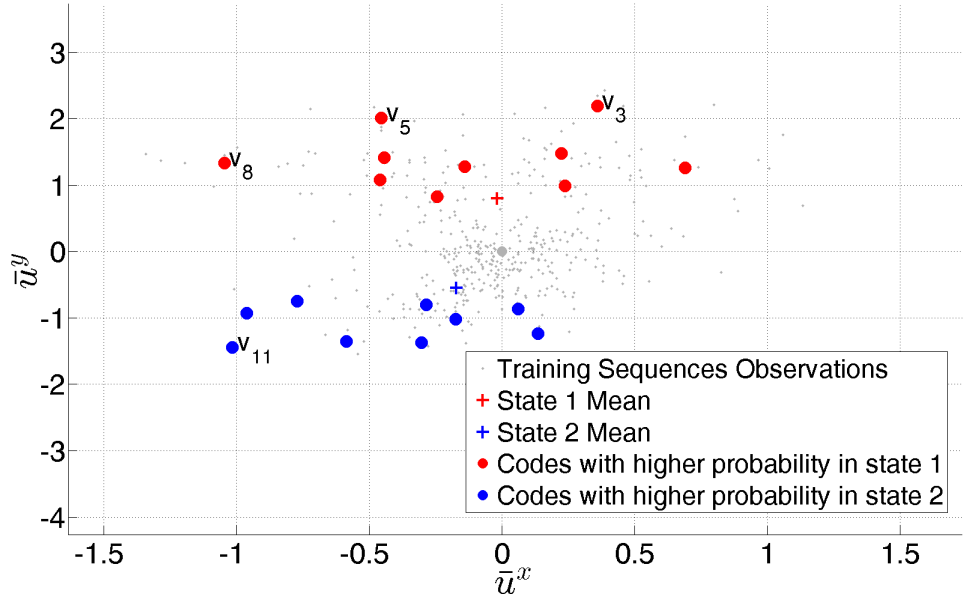


Figure 46. Training sequences and λ_3^{CPR} DHMM model parameters of cluster 3

4.3.4.5 Clusters models performance

In Figure 47, we plot the confidence of the training sequences in the four cluster models. Recall that, the confidence of a sequence in each cluster is defined [1] as the difference between its likelihood in the CPR cluster model and its likelihood in the non-CPR cluster model. As expected, the highest confidences occur when testing the sequences of clusters 1 and 3 in their respective cluster models. Similarly, sequences belonging to clusters 2 and 4 have higher confidence values in cluster models 2 and 4. However, few sequences of cluster 1 have higher confidences in models 2 and 4 than in models 1 and 3. Those sequences are actually non-CPR sequences belonging to cluster 1 but were assigned high confidences by the non-CPR models..

Since the output of each cluster model is a confidence value, we can treat them as individual classifiers and evaluate their performance on the training data. Figure 48 shows the ROCs of each cluster model on the training data $\mathfrak{D}_{Trn}$.

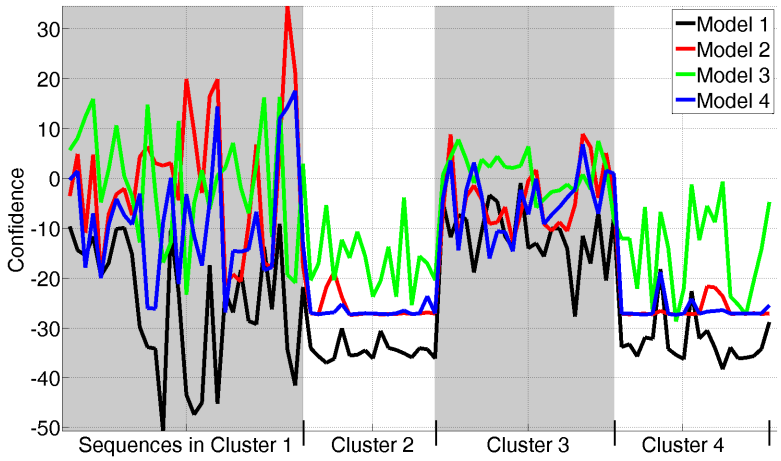
4.3.4.6 eHMM fusion results

In the previous steps of the eHMM, we initialized and learned multiple models for the different clusters. Each training sequence ($O_r \in \mathfrak{D}_{Trn}$) is tested in the K models and is assigned confidence value. Let $\mathbf{f}_r$ denote the K-dimensional vector of confidences assigned by λ_k to O_r. The multiple responses of the different models need to be combined into a single confidence value using one of the following combination methods.

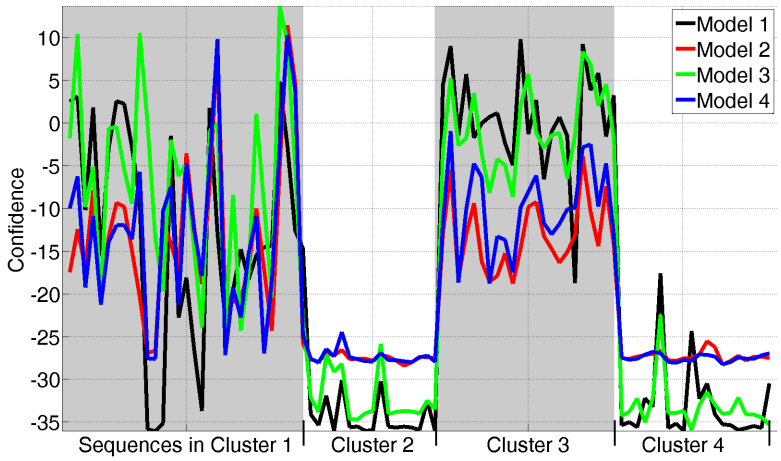
- Artificial Neural Network: A single layer perceptron with no hidden layers taking $\{\mathbf{f}_1, \ldots, \mathbf{f}_{R_n}\}$ as input. Using the standard backpropagation algorithm [64] , the ANN is trained to output the corresponding labels $\{y_1, \ldots, y_{R_n}\}$ of the training data. In other words, the ANN weights are optimized to minimize the misclassification error on the training data. In table 21, we show the weights of each node in the ANN's layer.

- Mixture of Experts: A one-level HME with branching factor of two is used. The input to the HME is a (K+1)-dimensional vector of the cluster models confidences plus an intercept. The HME parameters are updated using the expectation-maximization [26] algorithm and the labeled training data. These parameters are the gating network weights, $\mathbf{v_0}$, at the non-terminal node and the experts weights, $\mathbf{v_{11}}$ and $\mathbf{v_{12}}$, at the leaf nodes. The EM-learned parameters are reported in table 22.

[1] refer to equation (101) in section 4.2.4.1 for the formal definition.



(a) Using initial models



(b) Using BW-trained models

Figure 47. Confidence of the training sequences of each cluster in all cluster models

- Algebraic methods: We simply use algebraic operations such as the average, or the maximum to combine the confidences outputted by the K models. This method has no parameters and is not trainable. In this case, the extra step of computing the confidences of the training data is not needed.

Figure 49 shows the ROCs of the ANN, HME, and SUM fusion using the training data. To evaluate the fusion results, we juxtapose the ROC of the best cluster model classifier (cluster model 1 from figure 48). We notice that the SUM fusion performs poorly since it just averaged the confidence assigned by cluster 1 model with the confidences assigned by the other weaker clusters

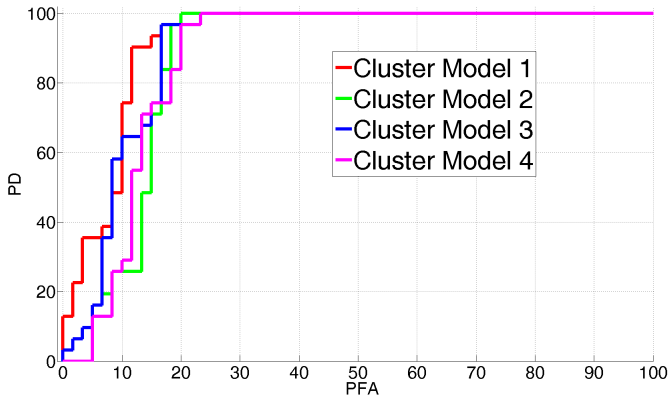


Figure 48. ROCs generated by the clusters models using the training data

TABLE 21

Weights and bias of the single-layer ANN after backpropagation training

Model	Weight
1	0.28
2	0.75
3	0.91
4	-0.26
Bias	-0.3

(2, 3, and 4) models classifiers. The ANN fusion ROC is slightly better than the best cluster model classifier. The HME outperforms significantly the other fusion methods and the cluster 1 model classifier. However this could be due to overfitting. The shape of the HME ROC suggests that the HME output is overfit to the data and tends to be discrete (0 for most of the training non-CPR sequences and 1 for most of the training CPR sequences).

4.3.4.7 eHMM performance

Recall that we use a 4-fold cross validation setting. For the n^{th} fold, a subset $\mathfrak{D}^n_{Trn}$, of size R_n, is used for training and another subset $\mathfrak{D}^n_{Tst}$, of size $Q_n = N - R_n$, is used for testing. All the eHMM steps detailed in the previous sections are repeated using the subset $\mathfrak{D}^n_{Trn}$ for each cross validation fold. The training step results in the mixture models $\{\lambda^{(n)}_k\}^K_{k=1}$, and an ANN- or HME-based fusion model, $\mathbb{H}_n$, for each crossvalidation fold n, $n = 1, \cdots, 4$. A new sequence $O_q \in \mathfrak{D}^n_{Tst}$ will be tested in the models $\{\lambda^{(n)}_k\}$. Depending on the type and the contents of the clusters, one or more models may have strong response to the test sequence O_q. The multiple responses of the different models will be combined into a single confidence value using $\mathbb{H}_n$. The label of O_q, CPR

TABLE 22

HME gating network and experts network parameters after EM training

Model	$\mathbf{v_0}$	$\mathbf{v_{11}}$	$\mathbf{v_{12}}$
1	4.16	-0.33	7.6
2	5.20	0.07	41.23
3	5.62	-0.04	-5.83
4	-2.26	0.03	-13.98
Intercept	190.00	-0.96	368.64

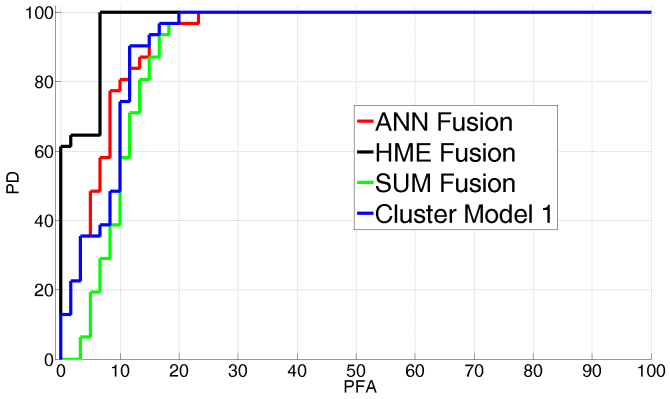


Figure 49. ROCs of the fusion methods and the best cluster model (from figure 48) using the training data

or non-CPR, is used only as a ground-truth to evaluate the performance of the eHMM. In figure 50, we report the ROCs of the eDHMM using ANN and HME fusion, and the ROC of the baseline DHMM ROCs provided earlier in figure 33 in section 4.3.3. The ROCs, generated using the same 4-fold crossvalidation partition, show that the eDHMM outperforms the baseline DHMM and that the ANN fusion method is better suited for the eDHMM. This may be due to the HME optimization being overfit to the training data (refer to figure 49).

In figure 51, we compare the ROCs of the eDHMM using ANN and HME fusion to the ROC of the baseline CHMM provided earlier in figure 36 in section 4.3.3. These ROCs show that the eCHMM outperfoms the baseline CHMM. However, the improvement of the eCHMM over the baseline CHMM is not a significant as in the discrete case. This can be seen in table 23, where we repot the Are Under Curve of all the ROCs of figures 50 and 51. Overall, we can conclude that the continuous eHMM is better suited for the data at hand, i.e identifying CPR sequences in video simulating medical crises using motion-based features.

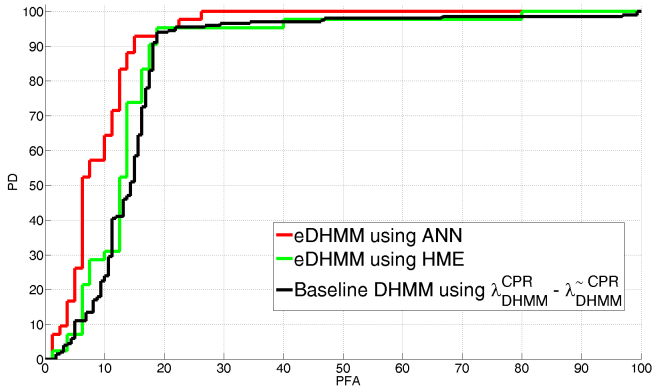


Figure 50. ROCs of the eDHMM and the baseline DHMM classifiers using 4-fold crossvalidation

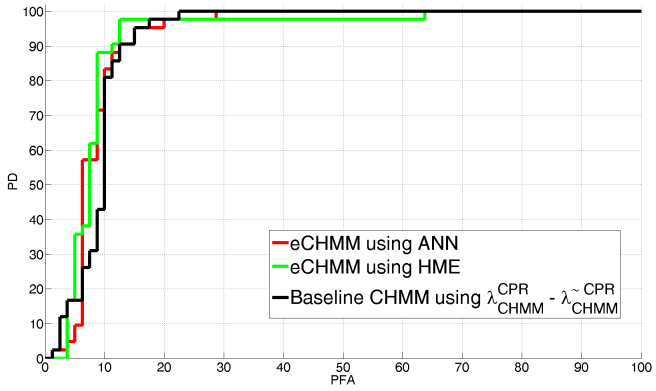


Figure 51. ROCs of the eCHMM and the baseline CHMM classifiers using 4-fold crossvalidation

4.4 Chapter summary

In this chapter, the proposed eHMM classifier is applied and evaluated on the classification of CPR sequences . First, we gave an overview of the preprocessing and extraction of CPR sequences from the simulation videos. Then, we outlined the architecture of the eHMM classifier and its intermediate steps. Finally, we compared the classification results of the eHMM to the baseline HMM. The experiments show that the proposed eHMM intermediate results are inline with the expected behavior and that, overall, the ensemble HMM outperforms the two-model baseline HMM. The results show also that both the HME and ANN fusion methods outperform the individual cluster models and simple combination methods.

TABLE 23

Area Under ROC Curve (AUC) of the eHMM and the baseline HMM classifiers

Classifier	AUC
eDHMM using ANN	538
eDHMM using HME	293
bDHMM	222
eCHMM using ANN	598
eCHMM using HME	611
bCHMM	574

CHAPTER 5

APPLICATION TO LANDMINE DETECTION

In this chapter, the proposed ensemble HMM classifier is evaluated using real datasets for landmine detection. The proposed eHMM learning steps are individually analyzed and, ultimately, the eHMM classifier's performance is compared against the baseline HMM classifier and other state-of-the-art algorithms.

5.1 Introduction

Detection, localization, and subsequent neutralization of buried antipersonnel (AP) and antitank (AT) landmines is a worldwide humanitarian and military problem. The latest statistics [77] show that in 2012, a total of $3,638$ casualties from mines were recorded in 62 countries and areas, including $1,066$ people killed and $2,552$ injured. In fact, the vast majority (78%) of recorded landmine casualties were civilians. Detection and removal of landmines is therefore a significant problem, and has attracted several researchers in recent years. One challenge in landmine detection lies in plastic or low metal mines that cannot or are difficult to detect by traditional metal detectors. Varieties of sensors have been proposed or are under investigation for landmine detection. The research problem for sensor data analysis is to determine how well signatures of landmines can be characterized and distinguished from other objects under the ground using returns from one or more sensors. Ground Penetrating Radar (GPR) offers the promise of detecting landmines with little or no metal content. Unfortunately, landmine detection via GPR has been a difficult problem [78, 79]. Although systems can achieve high detection rates, they have done so at the expense of high false alarm rates. The key challenge to mine detection technology lies in achieving a high rate of mine detection while maintaining low level of false alarms. The performance of a mine detection system is therefore commonly measured by a receiver operating characteristics (ROC) curve that jointly specifies rate of mine detection and level of false alarm.

Automated detection algorithms can generally be broken down into four phases: pre-processing, feature extraction, confidence assignment, and decision-making. Pre-processing algorithms perform

tasks such as normalization of the data, corrections for variations in height and speed, removal of stationary effects due to the system response, etc. Methods that have been used to perform this task include wavelets and Kalman filters [80], subspace methods and matching to polynomials [81], and subtracting optimally shifted and scaled reference vectors [82]. Feature extraction algorithms reduce the pre-processed raw data to form a lower-dimensional, salient set of measures that represent the data. Principal component (PC) transforms are a common tool to achieve this task [83, 84]. Other feature analysis approaches include wavelets [85], image processing methods of derivative feature extraction [86], curve analysis using Hough and Radon transforms [87], as well as model-based methods. Confidence assignment algorithms can use methods such as Bayesian [87], hidden Markov Models [86, 88, 33], fuzzy logic [89], rules and order statistics [90], neural networks, or nearest neighbor classifiers [91, 92], to assign a confidence that a mine is present at a point. Decision-making algorithms often post-process the data to remove spurious responses and use a set of confidence values produced by the confidence assignment algorithm to make a final mine/no-mine decision.

In [86, 88], hidden Markov modeling was proposed for detecting both metal and nonmetal mine types using data collected by a moving-vehicle-mounted GPR system and has proved that HMM techniques are feasible and effective for landmine detection. The initial work relied on simple gradient edge features. Subsequent work used an edge histogram descriptor (EHD) approach to extract features from the original GPR signatures. The baseline HMM classifier consists of two HMM models, one for mine and one for background. The mine (background) model captures the characteristics of the mine (background) signatures. The model initialization and subsequent training are based on averaging over the training data corresponding to each class.

5.2 Data preprocessing and pre-screening

5.2.1 GPR data

The input data consists of a sequence of raw GPR signatures collected by a NIITEK Inc. landmine detection system comprising a vehicle-mounted GPR array [3] (see figure 52). The NIITEK GPR collects 51 channels of data. Adjacent channels are spaced approximately five centimeters apart in the cross-track direction, and sequences (or scans) are taken at approximately five centimeter down-track intervals. The system uses a V-dipole antenna that generates a wide-band pulse ranging from 200 MHz to 7 GHz. Each A-scan, that is, the measured waveform that is collected in one channel at one down-track position, contains 416 time samples at which the GPR signal return is

Figure 52. NIITEK vehicle mounted GPR system [3]

recorded. Each sample corresponds to roughly 8 picoseconds. We often refer to the time index as depth although, since the radar wave is traveling through different media, this index does not represent a uniform sampling of depth. Thus, we model an entire collection of input data as a three-dimensional matrix of sample values, $S(z, x, y), z = 1, \cdots, 416; x = 1, \cdots, 51; y = 1, \cdots, N_S$, where N_S is the total number of collected scans, and the indices z, x,and y represent depth, cross-track position, and down-track positions respectively. A collection of scans, forming a volume of data, is illustrated in figure 53.

Figure 54 displays several B-scans (sequences of A-scans) both down-track (formed from a time sequence of A-scans from a single sensor channel) and cross-track (formed from each channel's response in a single sample). The surveyed object position is highlighted in each figure. The objects scanned are (a) a high-metal content antitank mine, (b) a low-metal antipersonal mine, and (c) a wood block.

5.2.2 Data preprocessing

Preprocessing is an important step to enhance the mine signatures for detection. In general, preprocessing includes ground-level alignment and signal and noise background removal. First, we identify the location of the ground bounce as the signal's peak and align the multiple signals with respect to their peaks. This alignment is necessary because the vehicle-mounted system cannot maintain the radar antenna at a fixed distance above the ground. The early time samples of each

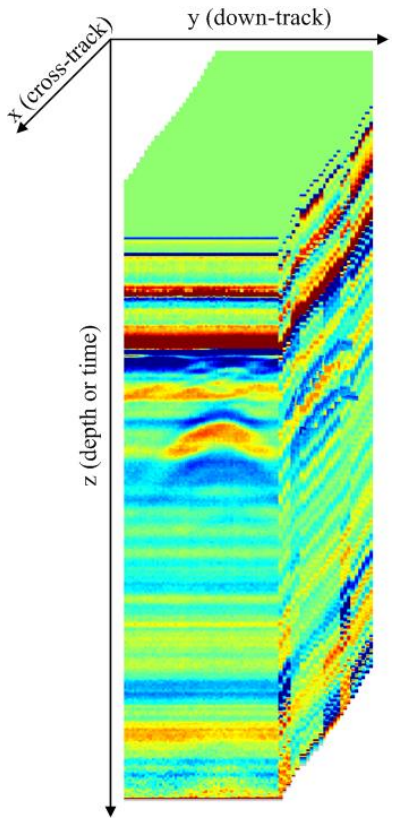
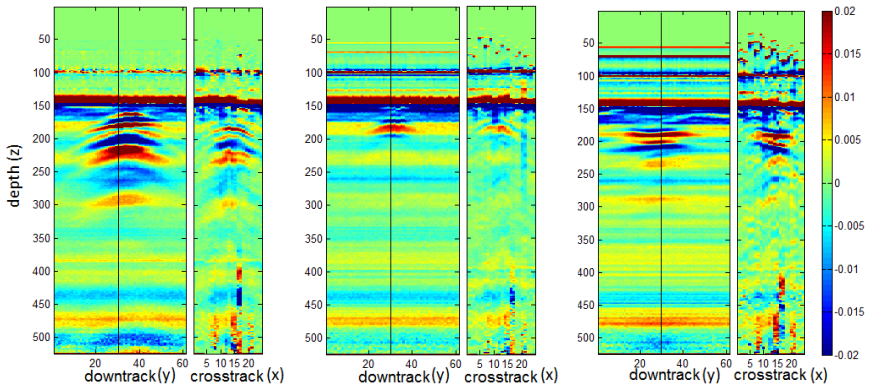


Figure 53. A collection of few GPR scans



(a) AT high-metal mine (b) AP low-metal mine (c) non-metal clutter

Figure 54. NIITEK radar down-track and cross-track (at position indicated by a line in the down-track) B-scans pairs for (a) an Anti-Tank (AT) mine, (b) an Anti-Personnel (AP) mine, and (c) a non-metal clutter alarm.

signal, up to few samples beyond the ground bounce are discarded. The remaining signal samples are divided into N depth bins, and each bin would be processed independently. The reason for this segmentation is to compensate for the high contrast between the responses from deeply buried and shallow anomalies. Next, the adaptive least mean squares (LMS) pre-screener [93] is used to focus attention and identify regions with subsurface anomalies. The goal of a pre-screener algorithm, in the framework of vehicle-mounted realtime landmine detection, is to flag locations of interest utilizing a computationally inexpensive algorithm so that more advanced feature-processing approaches can be applied only on the small subsets of data flagged by the pre-screener. The LMS is applied to the energy at each depth bin and assigns a confidence value to each point in the (cross-track, down-track) plane based on its contrast with a neighboring region. The components that satisfy empirically predetermined conditions are considered as potential targets. Their cross-track x_s, and down-track y_s positions of the connected component center are reported as alarm positions for further processing by the feature-based discrimination algorithm to attempt to separate mine targets from naturally occurring clutter.

5.3 Feature extraction

The goal of the feature extraction step is to transform original GPR data into a sequence of observation vectors. We use four types of features that have been proposed and used independently. Each feature represents a different interpretation of the raw data and aims at providing a good discrimination between mine and clutter signatures. These features are outlined in the following subsections.

5.3.1 EHD features

The Edge Histogram Descriptors (EHD) [94] captures the salient properties of the 3-D alarms in a compact and translation-invariant representation. This approach, inspired by the MPEG-7 EHD [95], extracts edge histograms capturing the frequency of occurrence of edge orientations in the data associated with a ground position. The basic MPEG-7 EHD has undergone rigorous testing and development, and thus, represents one of the generic and efficient texture descriptors. For a generic image, the EHD represents the frequency and the directionality of the brightness changes in the image. Simple edge detector operators are used to identify edges and group them into five categories: vertical, horizontal, 45° diagonal, 135° antidiagonal, and isotropic (nonedges). The EHD would include five bins corresponding to the aforementioned categories. For our application, we

adapt the EHD to capture the spatial distribution of the edges within a 3-D GPR data volume. To keep the computation simple, we still use 2-D edge operators. In particular, we fix the cross-track dimension and extract edges in the (depth, down-track) plane. The overall edge histogram is obtained by averaging the output of the individual (depth, down-track) planes. Also, since vertical, horizontal, diagonal, and antidiagonal edges are the main orientations present in the mine signatures, we keep the five edge categories of the MPEG-7 EHD.

Let $S_{zy}^{(x)}$ be the x^{th} plane of the 3-D signature $S(x, y, z)$. First, for each $S_{zy}^{(x)}$, we compute four categories of edge strengths: vertical, horizontal, 45° diagonal, 135° antidiagonal. If the maximum of the edge strengths exceeds a certain preset threshold θ_G, the corresponding pixel is considered to be an edge pixel. Otherwise, it is considered a nonedge pixel.

In the HMM models, we take the down-track dimension as the time variable (i.e., y corresponds to time in the HMM model). Our goal is to produce a confidence that a mine is present at various positions, (x, y), on the surface being traversed. To fit into the HMM context, a sequence of observation vectors must be produced for each point. The observation sequence of $S_{zy}^{(x)}$ at a fixed depth z, is the sequence of T observation vectors $H_{zy_i}^{(x)}$, $i = 1, \cdots, T$, each represents a five-bin edge histogram correspondent to $S_{zy_i}^{(x)}$.

The overall sequence of observation vectors computed from the 3-D signature $S(x, y, z)$ is then:

$$O(x, y, z) = [\overline{H}_{zy_1}, \overline{H}_{zy_2}, \cdots, \overline{H}_{zy_T}], \tag{105}$$

where $\overline{H}_{zy_i}$ is the cross-track average of the edge histograms of subimage $S_{zy_i}^{(x)}$ over N_C channels, i.e.

$$\overline{H}_{zy_i} = \frac{1}{N_C} \sum_{x=1}^{N_C} H_{zy_i}^{(x)}. \tag{106}$$

The extraction steps of the EHD features are illustrated in Figure 55.

Figures 56 and 57 display the edge histogram features for a strong mine and a clutter encounter identified by the pre-screener due to its high-energy contrast. As it can be seen, the EHD of the mine encounter can be characterized by a strong response of the diagonal and anti-diagonal edges. Moreover, the frequency of the diagonal edges is higher than the frequency of the anti-diagonal edges on the left side of the image (rising edge of the signature) and lower on the right part (falling edge). This feature is typical in mine signatures. The EHD features of the clutter encounter, on the other hand, does not follow this pattern. The edges do not follow a specific structure, and the diagonal and anti-diagonal edges are usually weaker.

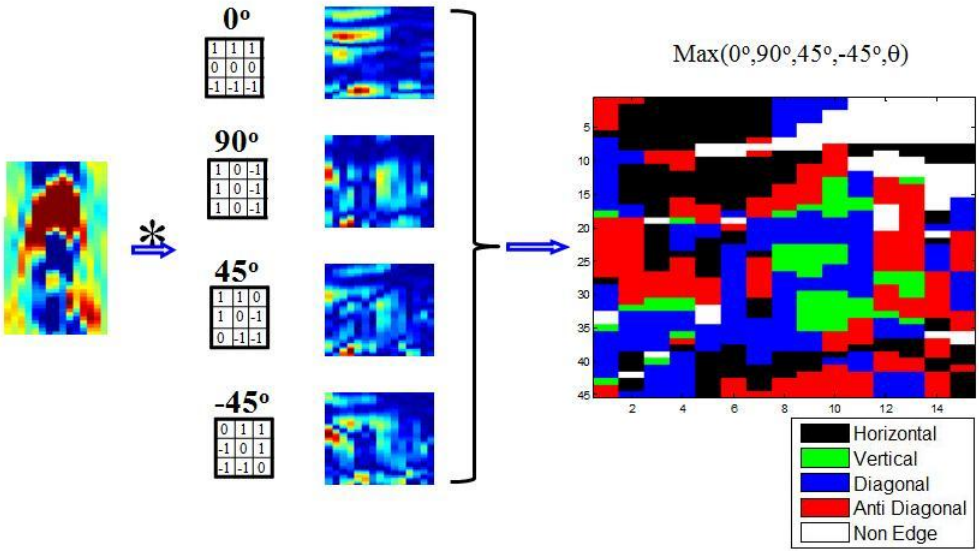
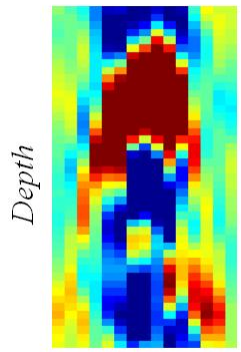


Figure 55. Illustration of the EHD feature extraction process.

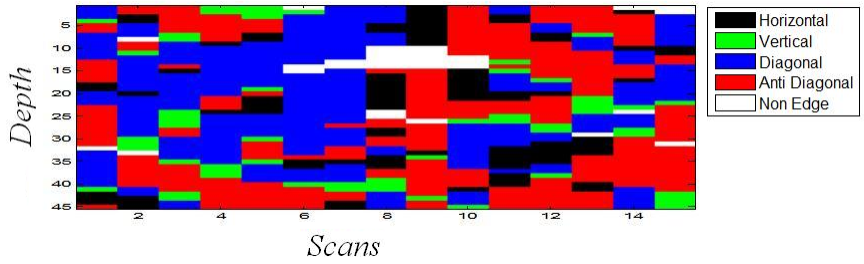
5.3.2 Gabor features

Gabor features characterize edges in the frequency domain at multiple scales and orientations and are based on Gabor wavelets [33]. This feature is extracted by expanding the signature's B-scan (depth vs. downtrack) using a bank of scale and orientation selective Gabor filters. Expanding a signal using Gabor filters provides a localized frequency description. In our experiments, without loss of generality, we use a bank of filters tuned to the combination of two scales, at octave intervals, and four orientations, at 45-degree intervals.

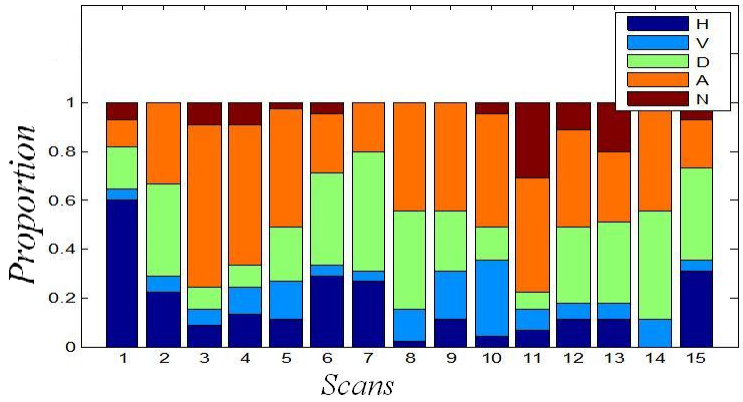
Let $S_x(y,z)$ be the x^{th} plane of the $3-D$ signature $S(x,y,z)$. Let $SG^{(k)}(x,y,z)$, $k=1,,8$ denote the response of $S_x(y,z)$ to the eight Gabor filters. Figure 58 displays the response of three distinct signatures to the eight Gabor filters. Figure 58(a) shows the response to a strong mine signatures. For this alarm, the signature has a strong response to the 45-degree filters (at both scales) on the left part of the signature (rising edge), and a strong response to the 135-degree filters on the right part of the signature (falling edge). Similarly, the middle of signature has a strong response to the horizontal filters (flat edge). Figure 58(b) displays the response of a weak mine signature. For this signature, the edges are not as strong as those in Figure 58(a). As a result, it has a weaker response at both scales, especially for the rising edge. Figure 58(c) displays a clutter signature (identified by the pre-screener) and its response. As it can be seen, this signature has strong response to the 45-degree filters. However, this response is not localized on the left side of the signature, and is not followed by a falling edge as it is the case for most mine signatures.



(a)

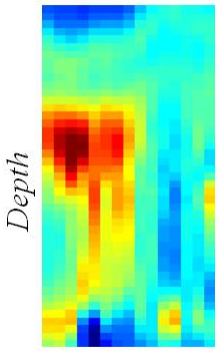


(b)

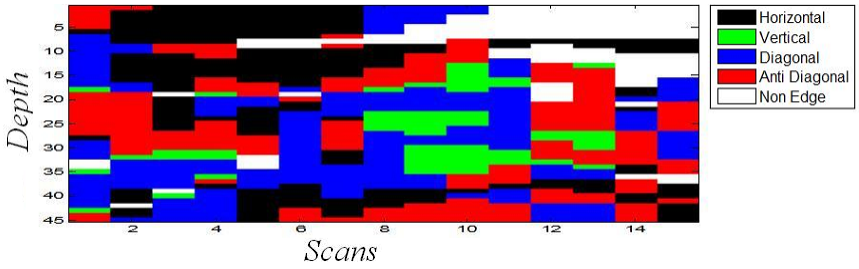


(c)

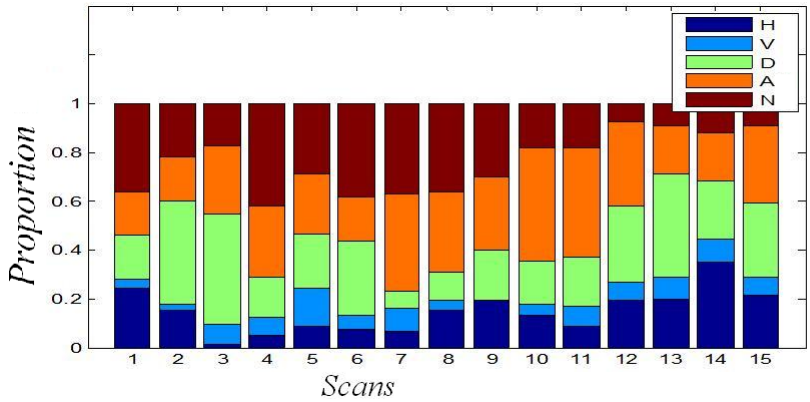
Figure 56. EHD features of a strong mine signature: (a) Mine signature in the (depth, down-track) plane, (b) Edge orientation of each location of the signature, (c) EHD features for the 15 observations



(a)

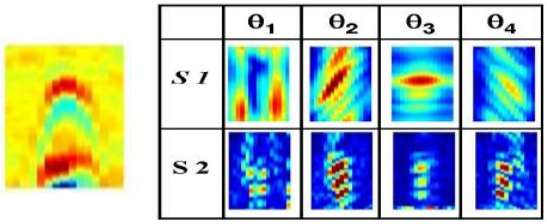


(b)

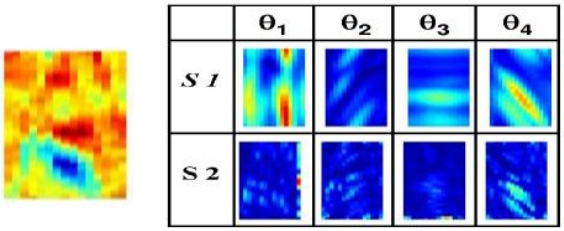


(c)

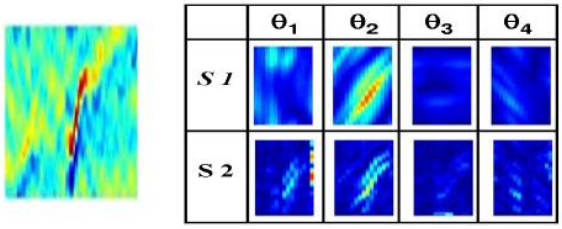
Figure 57. EHD features of a clutter encounter: (a) clutter GPR signature in the (depth, down-track) plane, (b) edge orientation of each location of the signature, (c) EHD features for the 15 observations



(a) Strong mine signature



(b) Weak mine signature



(c) Clutter signature

Figure 58. Response of three distinct alarm signatures to Gabor filters at two scales and four orientations.

The observation sequence of $S_x(y,z)$ at a fixed depth z, is the sequence of observation vectors:

$$O(x,y-p,z), O(x,y-p+1,z), \ldots, O(x,y-1,z), O(x,y,z), O(x,y+1,z), ..., O(x,y+p,z), \quad (107)$$

where

$$O(x,y,z) = [O^1(x,y,z), \ldots, O^8(x,y,z)] \quad (108)$$

and

$$O^k(x,y,z) = \sum_{z \in w} SG_x^{(k)}(y,z) \quad (109)$$

In (109), $O^k(x,y,z)$, encodes the response of $S(x,y,z)$ to the k^{th} Gabor filter, and w is a depth window around the depth being considered.

5.3.3 Gradient features

This feature encodes the degree to which edges occur in the diagonal and anti-diagonal directions within a B-scan [86]. First, we compute the first and second derivative on the raw signal $S(x, y, z)$ along the down-track (y) direction to remove the stationary effects and accentuate the edges in the diagonal and antidiagonal directions using:

$$\begin{aligned} D_y(x, y, z) &= \frac{[S(x, y+2, z) + 2S(x, y, z) - 2S(x, y-1, z) - S(x, y-2, z)]}{3}, \\ D_{yy}(x, y, z) &= \frac{[D_y(x, y+2, z) + 2D_y(x, y-1, z) - D_y(x, y-2, z)]}{3} \end{aligned} \tag{110}$$

Then, the derivative values are normalized using:

$$N(x, y, z) = \frac{D_{yy}(x, y, z) - \mu(x, z)}{\sigma(x, z)}, \tag{111}$$

where $\mu(x, z)$ and $\sigma(x, z)$ are the running mean and standard deviation updated using a small background area around the target flagged by the prescreener.

The down-track dimension is taken as the time variable in the HMM model. The observation vector at a point (x_s, y_s) consists of a set of 15 features that are computed on a normalized array of GPR data of size 32×8. For a given x_s and y_s, let $A = A(y, z) = N(x, y, z)$, where $x = x_s, y = y_s - 3, \ldots, y_s + 4$, and $z = 1, 2, \ldots, 32$. The array A is then broken into positive and negative parts according to the formulas:

$$A^+(y, z) = \begin{cases} A(y, z) & \text{if } A(y, z) \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{112}$$

$$A^-(y, z) = \begin{cases} -A(y, z) & \text{if } A(y, z) \leq -1 \\ 0 & \text{otherwise.} \end{cases} \tag{113}$$

Next, for each point in the positive and negative parts of A, the strengths of the diagonal and anti-diagonal edges are estimated. The strengths are measured by taking the local minimum in either the 45° or 135° direction around the column $y_s + 1$. Four types of edges that correspond to the, positive anti-diagonal (PA), negative anti-diagonal (NA), positive diagonal (PD), and negative diagonal (ND) edges are defined. These edges are computed using

$$\begin{aligned} PA(z) &= \min A^+(y_s, z-1), A^+(y_s+1, z), A^+(y_s+2, z+1), A^+(y_s+3, z+2) \\ NA(z) &= \min A^-(y_s, z-1), A^-(y_s+1, z), A^-(y_s+2, z+1), A^-(y_s+3, z+2) \\ PD(z) &= \min A^+(y_s, z+2), A^+(y_s+1, z+1), A^+(y_s+2, z), A^+(y_s+3, z-1) \\ ND(z) &= \min A^-(y_s, z+2), A^-(y_s+1, z+2), A^-(y_s+2, z), A^-(y_s+3, z-1) \end{aligned}$$

For each edge type, we find the position of the maximum value over a neighborhood of 32 depth values. For example, in the array PA we compute

$$m_{pa} = \text{argmax}\{PA(z) : z = 1, 2, \cdots, 32\} \tag{114}$$

where m_{pa} denotes "maximum of the positive anti-diagonal". The variables m_{pd}, m_{na}, and m_{nd} are defined similarly. The values of the positive and negative diagonal and anti-diagonal arrays are used to define the 4-dimensional (4-D) observation vector associated with the point (x_s, y_s), $O(x_s, y_s) = [PD(m_{pd}), PA(m_{pa}), ND(m_{nd}), NA(m_{na})]$. Observation sequences of length 15 are formed at point (x, y) by extracting the observation sequence:

$$O(x, y-7), O(x, y-6), \cdots, O(x, y-1), O(x, y), O(x, y+1), \cdots, O(x, y+7).$$

5.3.4 Bar features

The bar features have been first used in the context of handwritten character recognition [96]. Here, we adapt them to GPR data as follows. For each Ndepths-by-Nscans B-scan, the original image is normalized and broken into positive and negative sub-images. Let $S_{zy}^{(x)}$ be the x^{th} plane of the 3-D signature $S(x, y, z)$. Then, each part of S is binarized using a constant threshold. Eight feature images ($H^+(z, y)$, $D^+(z, y)$, $V^+(z, y)$, $A^+(z, y)$, $H^-(z, y)$, $D^-(z, y)$, $V^-(z, y)$, and $A^-(z, y)$) are generated. Each feature image corresponds to one of the directions: Horizontal, Diagonal, Vertical, and Antidiagonal in either the positive or negative binarized image. Each feature image has an integer value at each location that corresponds to the number of successive pixel values equal to 1 (or -1 for the negative images) in that direction. These numbers are computed efficiently using the two-pass procedure described in Algorithm 7. The feature image in each direction is obtained by simple summation of the feature images for the positive and negative sub-images, using :

$$\begin{cases} H(z, y) = H^+(z, y) + H^-(z, y) \\ D(z, y) = D^+(z, y) + D^-(z, y) \\ V(z, y) = V^+(z, y) + V^-(z, y) \\ A(z, y) = A^+(z, y) + A^-(z, y). \end{cases}$$

An example of an original GPR B-scan image and the derived bar features images is shown in figure 59. Each entry, in the matrix H for example, is given a value equal to the longest horizontal bar that fits in the horizontal direction around that entry. Thus, the bar feature matrices H, D, V, and A measure the strength of the horizontal, diagonal, vertical, and anti-diagonal edge at each point of the original GPR image.

Algorithm 7 Pseudo-code for computing the bar features

Require: 2-D matrix of binarized GPR image (Ndepths x Nscans).
Ensure:

{FORWARD PASS}
for $z = 2, \cdots, Ndepths - 1$ **do**
    for $y = 2, \cdots, Nscans - 1$ **do**
        H(z,y)=H(z,y-1)+1
        D(z,y)=D(z-1,y+1)+1
        V(z,y)=V(z-1,y)+1
        A(z,y)=A(z-1,y-1)+1
    end for
end for
{BACKWARD PASS}
for $z = Ndepths - 1, \cdots, 2$ **do**
    for $y = Nscans - 1, \cdots, 2$ **do**
        H(z,y)=max(H(z,y),H(z,y+1))
        D(z,y)=max(D(z,y),D(z+1,y-1))
        V(z,y)=max(V(z,y),V(z+1,y))
        A(z,y)=max(A(z,y),A(z+1,y+1))
    end for
end for

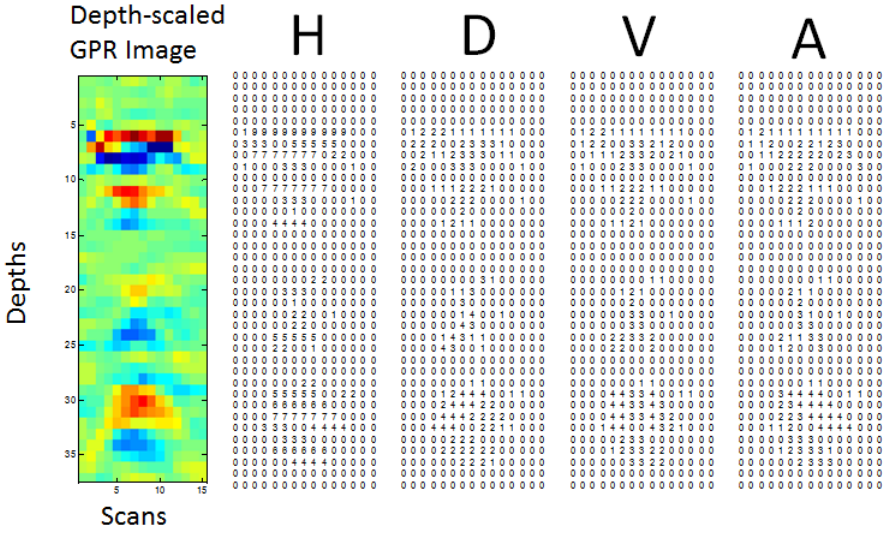


Figure 59. A mine B-scan image and the bar features matrices for the horizontal (H), diagonal (D), vertical (V), and anti-diagonal (A) directions

To derive meaningful features for HMM modeling, the feature images are combined into a 5-bin histogram: one bin for each direction and the 5^{th} for the non-edge direction. In particular, if the maximum of the bar matrices exceeds a certain preset threshold θ_B, the corresponding pixel is considered to be an edge pixel in the corresponding direction. Otherwise, it is considered a non-edge pixel. For a given channel $x \in \{1, \cdots, N_C\}$, the corresponding histogram, $\mathfrak{h}_{zy}^{(x)}$, is computed from the features matrices $\{H(z,y), D(z,y), V(z,y), A(z,y)\}$. The overall sequence of observation vectors computed from the 3-D signature $S(x,y,z)$ is then:

$$O(x,y) = [\overline{H}_{zy_1}, \overline{H}_{zy_2}, \cdots, \overline{H}_{zy_T}], \tag{115}$$

where $\overline{H}_{zy_i}$ is the cross-track average, N_C channels, of the bar features histograms:

$$\overline{H}_{zy_i} = \frac{1}{N_C} \sum_{x=1}^{N_C} \mathfrak{h}_{zy_i}^{(x)}. \tag{116}$$

5.4 Baseline HMM detector

The baseline HMM classifier consists of two HMM models, one for mine and one for background. Each model has four states and produces a probability value by backtracking through model states using the Viterbi algorithm [8]. The mine model, λ^m, is designed to capture the hyperbolic spatial distribution of the features. It uses observation vectors that encode the degree to which edges occur in the horizontal (H), vertical (V), diagonal (D), anti-diagonal (A), and non-edge (N) directions. This model assumes that mine signatures have a hyperbolic shape comprised of a succession of rising, horizonal, and falling edges with variable duration in each state. Eventually, the beginning and the end of the observation vectors correspond to non-edge (or background) state.

The 4-state mine model is illustrated in figure 60. In this case, the mine model is a cyclic model with the following architecture constraints:

- The system starts always at state 1;
- When in state i, the system can move only to state i or state $i+1$, $i = 1 \cdots N-1$;
- From state N, the system can move to state N or state 1.

The background model, λ^b, is needed to capture the background and clutter characteristics. No prior information or assumptions are used in this model.

The probability value produced by the mine (background) model can be thought of as an estimate of the probability of the observation sequence given that there is a mine (background) present. The model architecture is illustrated in figure 61.

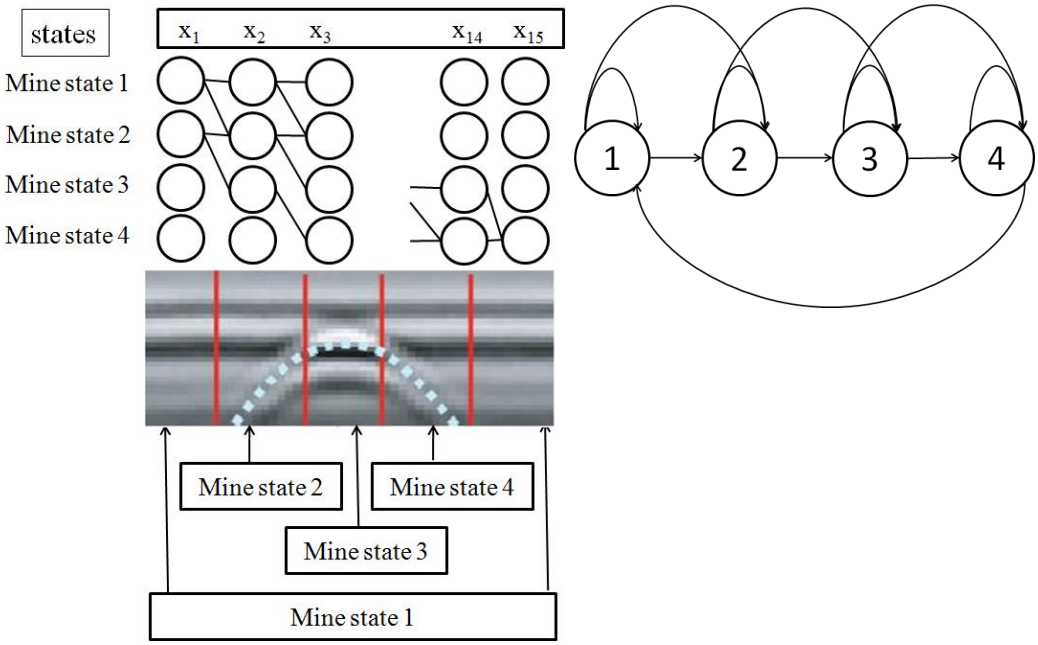


Figure 60. Illustration of the baseline HMM mine model with four states.

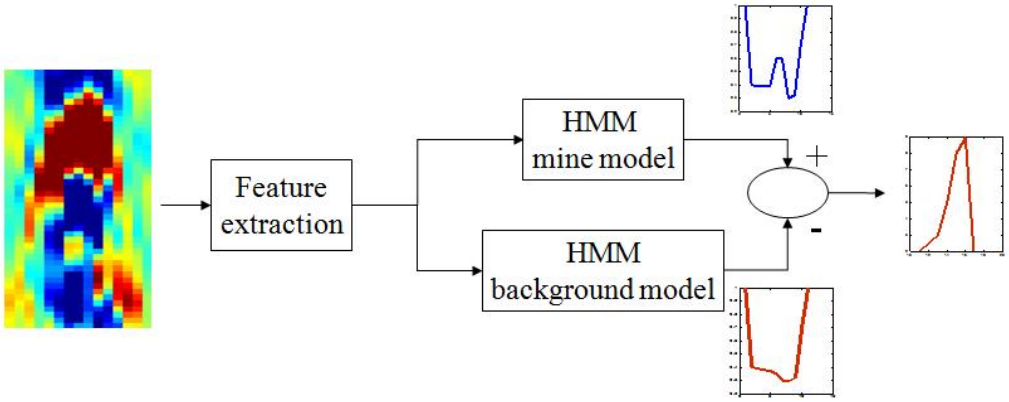


Figure 61. Illustration of the baseline HMM classifier architecture

To generate the codebook, we cluster the training data corresponding to each class (mine or background) into M clusters using the k-means algorithm [23]. The initial model parameters are set according to the distance of the M codebooks to each of the N states, using equation (69). The transition probabilities **A** and the observation probabilities **B** are then estimated using the Baum-Welch algorithm [9], the MCE/GPD algorithm [11], or a combination of the two.

The confidence value assigned to each observation sequence, Conf(O), depends on: (1) the probability assigned by the mine model, $Pr(O|\lambda^m)$; (2) the probability assigned by the background

model, $Pr(O|\lambda^c)$; and (3) the optimal state sequence. In particular, we use:

$$\text{Conf}(O) = \begin{cases} \max(\log \frac{Pr(O|\lambda^m)}{Pr(O|\lambda^c)}, 0) & \text{if } \#\{s_t = 1, t = 1, \cdots, T\} \leq T_{max} \\ 0 & \text{otherwise} \end{cases} \tag{117}$$

where $\#\{s_t = 1, t = 1, \cdots, T\}$ corresponds to the number of observations assigned to the background state (state 1). T_{max} is defined experimentally based on the shortest mine signature. Equation (117) ensures that sequences with a large number of observations assigned to state 1 are considered non-mines.

5.5 Ensemble HMM landmine detector

Mine signatures vary according to the mine type, mine size, and burial depth. Similarly, clutter signatures vary with soil type, site moisture, and other noise and environmental factors. The eHMM approach attempts to learn these variations by constructing an ensemble of HMMs that captures the characteristics and variability within the training data. In the following, we assume that the training data consists of a set of R sequences of length T.

The architecture of the eHMM landmine classifier is shown in figure 62. It has five main components namely, feature extraction, similarity matrix computation, similarity-based clustering, clusters' models construction, and decision level fusion. In the first step, the EHD, Gabor, gradient, and bar features are extracted as detailed in section 5.3. In the second step, each of the R sequences is used to learn one HMM model. The model parameters are initialized and subsequently adjusted, using the Baum-welch algorithm, to fit the corresponding sequence. The third step consists of building a pairwise similarity matrix based on the log-likelihood score and the Viterbi path mismatch penalty that result from testing each sequence in each model. The mixing factor α in (74) is set empirically to 0.1. In the fourth step, we use a standard hierarchical clustering algorithm to partition the data into groups (clusters) of similar signatures. Then, for each cluster, an HMM model is initialized and trained using the signatures belonging to that cluster. The devised training scheme depends on the cluster homogeneity and size. Finally, in the fifth step, we use a HME or ANN to combine the outputs of each model and assign a single confidence value to a given test sequence.

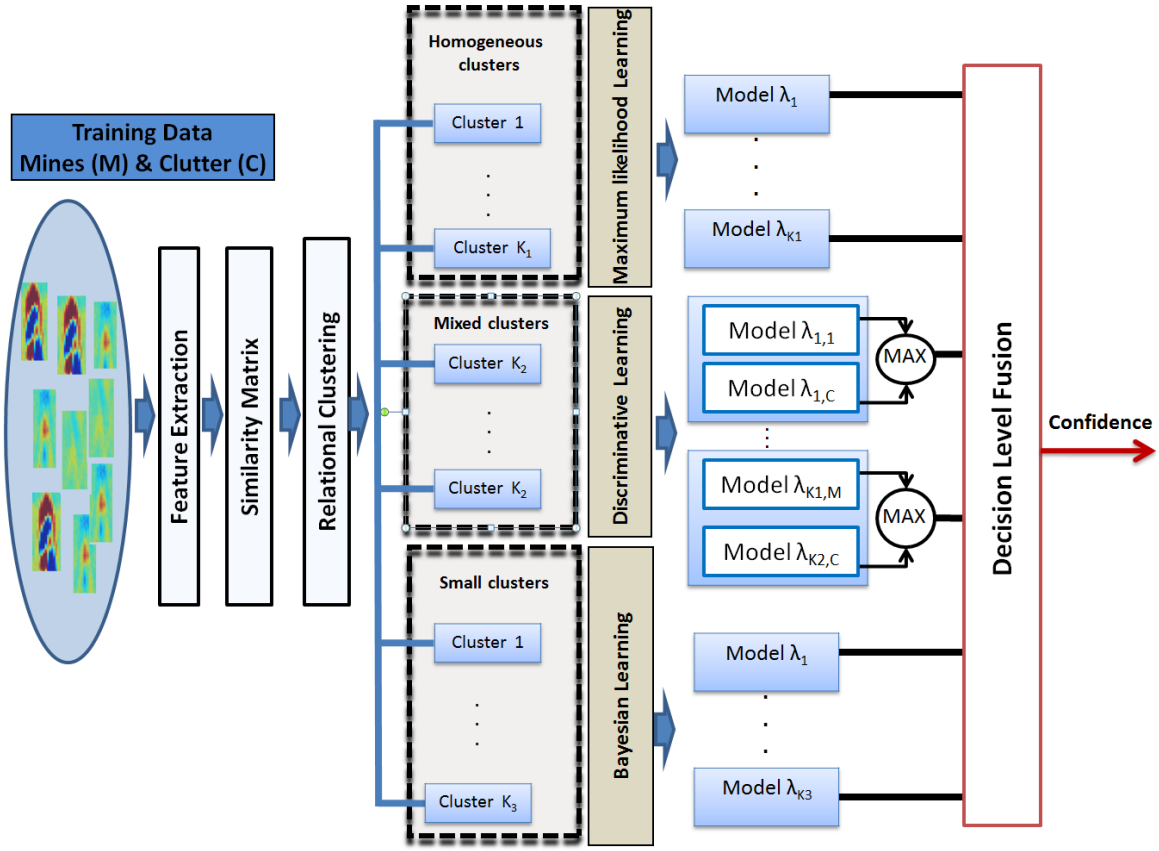


Figure 62. Block diagram of the proposed eHMM landmine classifier (training)

5.6 Experimental results

5.6.1 Dataset collections

The proposed eHMM was developed and tested on GPR data sets collected with a NIITEK vehicle mounted GPR system [3] (see figure 52). The datasets are comprised of a variety of mine and background signatures. We use data collected from outdoor test lanes at three different locations. The first two locations, site 1 and site 2, were temperate regions with significant rainfall, whereas the third collection, site 3, was a desert region. The lanes are simulated roads with known mine locations. Lanes at site 1 are labeled lanes 1, 3, and 4, and are 500 meters long and 3 meters wide. Lanes at site 2 are labeled lanes 3, 4, 13, 14, and 19, and are 50 to 250 meters long and 3 meters wide. Lanes at site 3 are labeled lanes 51 and 52, and are 300 meters long and 3 meters wide. Multiple data collections were performed at each site at different dates. Statistics of these datasets are reported in table 24. In the first dataset collection, $\mathfrak{D}_1$, the LMS pre-screener has identified a total of 1843 alarms, 613 of them are mine encounters. The second and third dataset collections,

$\mathfrak{D}_2$ and $\mathfrak{D}_3$ are obtained form the same GPR raw data, but using different preprocessing and prescreening methods. Based on the first preprocessing method, the prescreener used for $\mathfrak{D}_2$ identifies 724 mine encounters and 619 false alarms. With the second preprocessing method, used for $\mathfrak{D}_3$, the prescreener identifies 732 mine encounters, i.e. eight more mines than $\mathfrak{D}_2$, at the expense of 1126 additional false alarms.

TABLE 24

Data collections

	Total pre-screener alarms	Mine encounters	False alarms
$\mathfrak{D}_1$	1843	613	1230
$\mathfrak{D}_2$	1343	724	619
$\mathfrak{D}_3$	2477	732	1745

5.6.2 Data preprocessing and features extraction

5.6.2.1 Data preprocessing

For each dataset collection, the raw GPR data is preprocessed and prescreened, as detailed in section 5.2, to identify potential locations of interest within the GPR signals. Those locations, also called alarms, can be mine targets or naturally occurring clutter. The prescreener alarms, in the form of a 3-D GPR volume of data, are then used to extract the various features as described in section 5.3.

Each alarm has over 400 depth values, 51 channel samplings, and 61 scans. Experimental results show that we can reduce original data without affecting the overall performance. Thus we down-sample the GPR data along depth (by a factor of 4) and we process only 7 channels centered around the channel with maximum prescreener confidence. The prescreening step results in alarms that are centered around the middle scan position. Therefore, we use only the middle 15 scans $(24, \cdots, 38)$ of the original prescreener alarm. The first and last few scans are typically used for preprocessing (background variance estimation, background substraction, etc...). It is worth noting that the ground-truth depths of mine signatures are either manually identified or previously estimated by the prescreener. Those groundtruth depths are used only in the training step.

5.6.2.2 Features extraction

The previous step resulted in a smaller $7 \times 15 \times 100$ GPR volume. Consequently, each of the feature extraction methods, described in section 5.3, is used to extract a feature vector at

each down-sampled depth. Depending on the feature extraction method, averaging over the seven channels is done within the extraction procedure (for EHD) or post-extraction (for the Gabor, bar, and gradient features).

Given the downsampled volume of GPR data obtained in the previous step, we apply each of the extraction methods to compute the sequence of feature vectors at each depth. Each extraction method results in a map of 15 sequences of 4- or 5-dimensional feature vectors at each down-sampled depth. Let $\mathfrak{O}_l{}^{EHD}$, $\mathfrak{O}_l{}^{GBR}$, $\mathfrak{O}_l{}^{GRA}$, and $\mathfrak{O}_l{}^{BAR}$ denote the observation vector map extracted from dataset $\mathfrak{D}_l$, $\{l = 1, 2, 3\}$, using the EHD, Gabor, gradient, and bar features, respectively.

In figure 63, we show the observation vectors extracted from three signatures. The first (resp. second) signature corresponds to a mine with high (resp. low) metal content taken at the groundtruth depth and at the center channel identified by the prescreener. The third signature corresponds to a subwindow of an alarm taken at a random depth with no significant GPR signature activity. Note that the EHD and Gabor features are normalized, while the gradient and bar features values have wider ranges. The extracted feature vectors cover different aspects of the GPR signature. For instance, Gabor features are less prone to the intensity in the image and vary mainly with the size of the target and the texture of the signature.

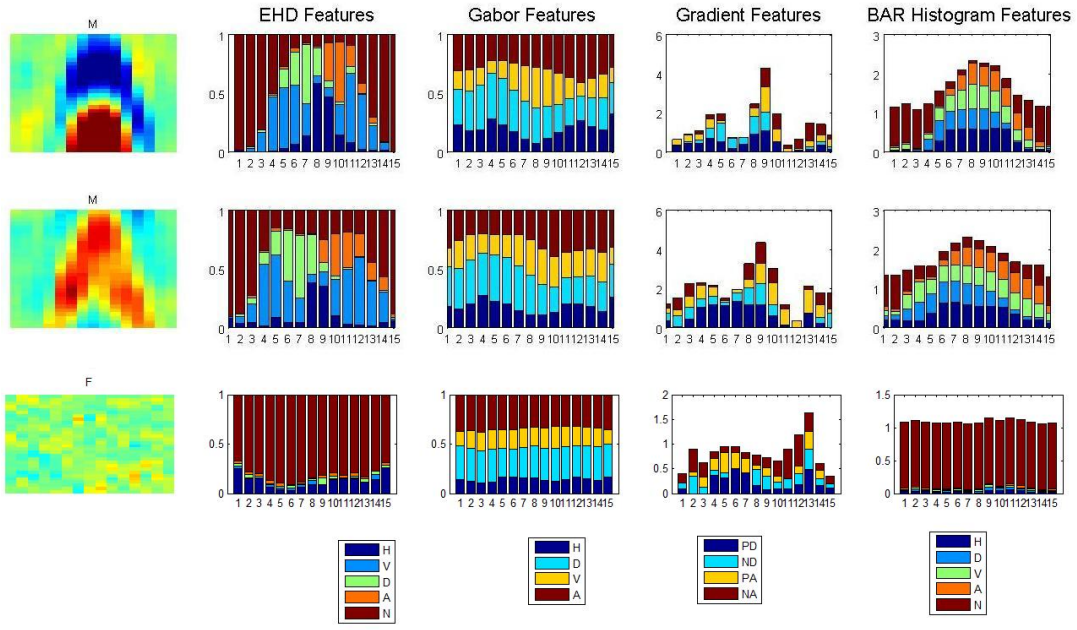
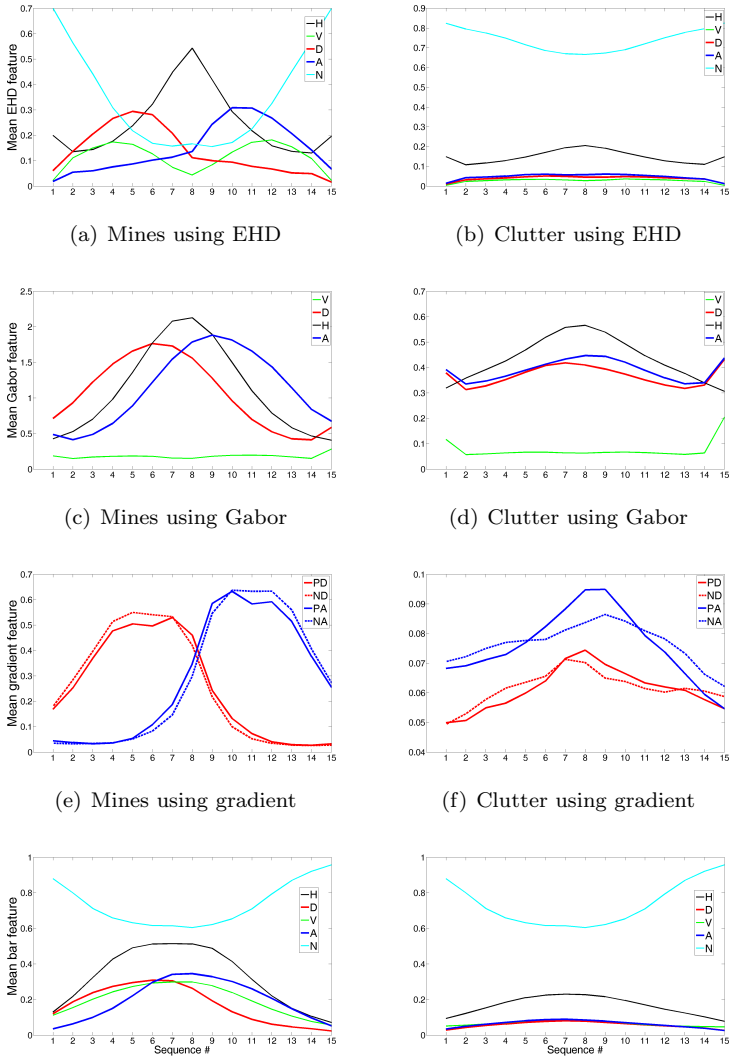


Figure 63. EHD, Gabor, gradient, and bar features of two sample mines and a background signature

In figure 64, we show the means of the extracted feature vector observations of all the



(a) Mines using EHD (b) Clutter using EHD

(c) Mines using Gabor (d) Clutter using Gabor

(e) Mines using gradient (f) Clutter using gradient

(g) Mines using bar (h) Clutter using bar

Figure 64. Average feature observation vectors of mines (column 1), and clutter (column 2) using EHD (row 1), Gabor (row 2), gradient (row 3), and bar extraction method (row 4)

mines (first column subfigures) and clutter (second column subfigures) of $\mathfrak{D}_1$. For each feature $Feat \in \{EHD, GBR, GRA, BAR\}$, we compute $\bar{O}^{(t)}_{Feat} = \textbf{average}\{O^{(t)} \in \mathfrak{O}_1{}^{Feat}\}$ for mines and clutter signatures separately. Then, the 5- or 4-dimensional mean vector of the EHD (row 1), Gabor (row 2), gradient (row 3), and bar features (row 4 sub-figures) is plotted at each observation position t, $1 \leq t \leq T$. These feature dependent global statistics could be used to set the thresholds of the proposed algorithms as outlined in the next section.

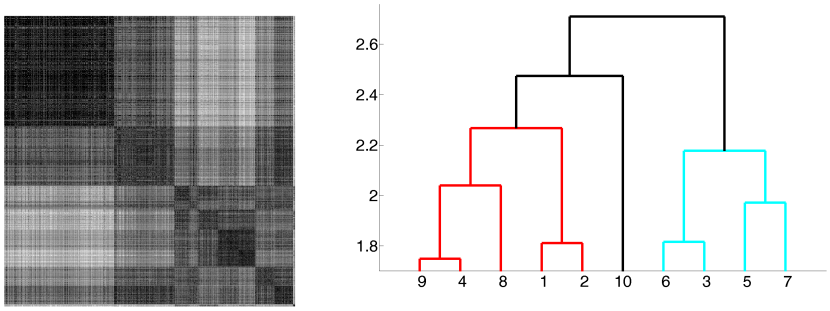
5.6.2.3 Identifying actual length of mine sequences

In order to satisfy the eHMM architecture constraints, we use equation (73) with the feature-dependent threshold τ to identify the actual length of the mine sequences. From figure 64(a), It can be seen that choosing $\tau = 0.18$, for the EHD features, would result in discarding a few mine observations with weak diagonal component from the start of the sequence and a few mine observations with weak anti-diagonal component at the end of the sequence. Similarly, for the other features and referring to figures 64(c), (e) and (g), respectively, we fix $\tau = 0.7$ for the Gabor features, $\tau = 0.3$ for the gradient features, and $\tau = 0.11$ for the bar features.

5.6.3 Experimental setup

In all experiments reported this chapter, we use a 6-fold cross validation for each data set $\mathfrak{D}_{\mathfrak{l}}$, $\{l = 1, 2, 3\}$. For each fold, a subset of the data ($\mathfrak{D}_{\mathfrak{l}Trn}$) is used for training and the remaining data ($\mathfrak{D}_{\mathfrak{l}Tst}$) is used for testing. Let R_n^l be the size of $\mathfrak{D}_{\mathfrak{l}Trn}$ and $Q_n^l = N^l - R_n^l$ be the size of $\mathfrak{D}_{\mathfrak{l}Tst}$ for the n^{th} fold. Finally let $\mathfrak{O}_{\mathfrak{l}Trn}^{Feat}$ denote the observation vector maps extracted from dataset $\mathfrak{D}_{\mathfrak{l}}$, $\{l = 1, 2, 3\}$, using the corresponding feature extraction method $Feat \in \{EHD, GBR, GRA, BAR\}$ ($Feat$ labels 'EHD', 'GBR', 'GRA', and 'BAR' correspond to the EHD, Gabor, gradient, and bar features respectively).

A comparison of the classifiers and the feature-based models should be based on the ROC curves and should use the same cross-validation settings. Therefore, we use a unified cross-validation experimental setup. For each dataset, we have the same crossvalidation partition, and the same setting for the different feature extraction methods. In particular, the EHD, Gabor, gradient, and bar features are pre-computed and saved for each dataset and will be used by all classifiers, namely the baseline DHMM, the baseline CHMM, the eDHMM, and the eCHMM. In the following results sections, we provide the implementation details of the eCHMM and eDHMM on one of the datasets $\mathfrak{D}_{\mathfrak{l}}$ and using one of the feature extraction methods seemingly. We illustrate some of the steps using $\mathfrak{D}_{\mathfrak{3}}$ and the eCHMM with bar features. In the last results section, we report the comprehensive results, in terms of area under ROC curve (AUC), of the eHMM vs the baseline HMM: (1) in the continuous vs. discrete case; (2) using EHD, Gabor, gradient, and bar features; and (3) using $\mathfrak{D}_{\mathfrak{1}}$, $\mathfrak{D}_{\mathfrak{2}}$, and $\mathfrak{D}_{\mathfrak{3}}$.



(a) (b)

Figure 65. eCHMM hierarchical clustering of $\mathfrak{O}_{3Trn1}^{BAR}$: (a) pairwise-distance matrix, (b) dendrogram

5.6.4 eHMM construction and training steps

For each dataset $\mathfrak{D}_l$, $l \in 1, 2, 3$, and each feature extraction method, $Feat \in \{EHD, GBR, GRA, BAR\}$, we have a 6-fold crossvalidation partition $\mathfrak{O}_l{}^{Feat}$=$\{\mathfrak{O}_{lTrn}^{Feat}, \mathfrak{O}_{lTst}^{Feat}\}$ of observation maps. Each alarm in the original GPR data is represented by a $T \times D$ sequence of feature vector observations at each down-sampled depth location.

The first step of the eHMM is the similarity matrix computation. This step requires fitting an individual HMM model for each sequence in the training data $\mathfrak{O}_{lTrn}^{Feat}$. Details of the individual DHMM construction and the similarity matrix computation of the first crossvalidation fold of dataset $\mathfrak{D}_3$ using the EHD features, $\mathfrak{O}_{3Trn1}^{EHD}$, were provided in the running example of chapter 3.

In the second step, the similarity matrix is transformed into a distance matrix, $\mathbf{D}$, using (77). The hierarchical clustering algorithm [24] is then applied, using $\mathbf{D}$ with a fixed number of clusters $K = 10$, to partition the training data. For both the discrete and continuous versions, using any of the features and dataset, the eHMM clustering step successfully assigns groups of similar alarms into clusters. For instance, in figures 65 and 66, we show the hierarchical clustering results of the first crossvalidation fold of the eCHMM using bar features on $\mathfrak{D}_3$. As it can be seen in figure 66(a) and in the dendrogram of figure 65(b), we have a group of clutter dominated clusters (in red) an a second group of clusters dominated by mines (in cyan). Additional details of the clusters' contents per mine type and per burial depth are shown in figures 66(b) and (c).

In the third step, we build the eHMM mixture models for the clusters obtained from the previous step. The eHMM is built based on the clusters' sizes and homogeneity. Then, the context dependent training scheme, described in section 3.3.3, is used to initialize and learn the eHMM. Continuing with the case example of the eCHMM with BAR features, $\mathfrak{D}_{3Trn1}^{BAR}$, of the previous step, we can see that the partition of the training data includes four homogeneous clusters (Clusters 6, 7,

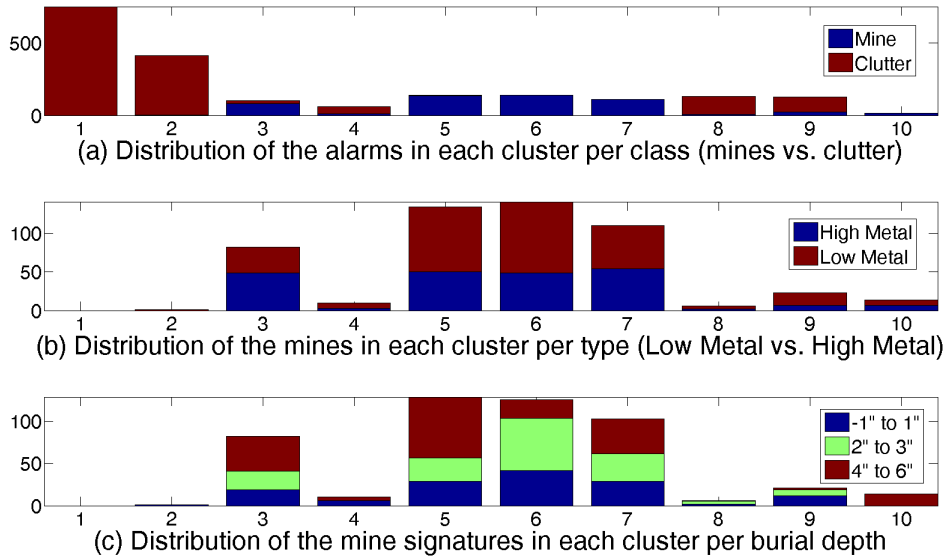


Figure 66. eCHMM hierarchical clustering results of $\mathfrak{O}_{3Trn1}^{BAR}$: distribution of the alarms in each cluster: (a) per class, (b) per type, (c) per depth.

and 10 contain only mines and cluster 1 has only clutter). The remaining clusters (2, 3, 4, 5, 8, and 9) are mixed. Therefore, using the notation of Algorithm 6, we define our eHMM as :

$$\begin{cases} \lambda_i^{BW} = \{\lambda_1^{(F)}, \lambda_6^{(M)}, \lambda_7^{(M)}, \lambda_{10}^{(M)}\}, \\ \lambda_j^{MCE} = \{\lambda_j^{(M)}, \lambda_j^{(C)}\}, j \in \{2,3,4,5,8,9\}, \\ \lambda_k^{VB} = \emptyset. \end{cases} \tag{118}$$

Table 25 shows the BW-trained eCHMM model for cluster 6, $\lambda_6^{(M)}$. Recall that cluster 6 contains only mine signatures that have a low metal content and/or buried at 2" or deeper, as it can be seen in figure 66. Therefore, the alarms in cluster 6 are expected to have weak signatures and therefore weak edge features. This could explain the large non-edge component of most of the states means components of $\lambda_6^{(M)}$ reported in table 25. Moreover, both states s_1 and s_3 have two dominant components (g_{11}, g_{12} for s_1; g_{32}, g_{33} for s_2) and one weak component. In both cases, the dominant components are the ones with the largest no-edge attribute. Nevertheless, the states representatives still characterize the hyperbolic shape of a typical mine signature, i.e. the succession of $Dg - Hz - Ad$ states. For instance, all s_1 components means have their diagonal D dimension larger than the anti-diagonal A dimension.

In the final step, the eHMM mixture constructed previously is combined using an ANN or a HME. The ANN and HME parameters are trained to fit the responses of the eHMM mixture models

TABLE 25

λ_6^M CHMM model parameters of cluster 6

A

	s_1	s_2	s_3
s_1	0.75	0.25	0.00
s_2	0.00	0.81	0.19
s_3	0.00	0.00	1.00

Priors

	π
s_1	1.00
s_2	0.00
s_3	0.00

Means

		H	D	V	A	N
s_1	c_{11}	0.20	0.21	0.16	0.05	0.82
	c_{12}	0.47	0.31	0.26	0.15	0.61
	c_{13}	0.60	0.27	0.34	0.20	0.36
s_2	c_{21}	0.55	0.24	0.25	0.31	0.57
	c_{22}	0.56	0.39	0.44	0.36	0.37
	c_{23}	0.67	0.30	0.38	0.43	0.27
s_3	c_{31}	0.52	0.14	0.32	0.36	0.41
	c_{32}	0.15	0.04	0.11	0.16	0.86
	c_{33}	0.36	0.09	0.23	0.31	0.66

Covariance

		H	D	V	A	N
s_1	cov_{11}	0.01	0.01	0.01	0.01	0.01
	cov_{12}	0.01	0.01	0.01	0.01	0.01
	cov_{13}	0.01	0.01	0.01	0.01	0.01
s_2	cov_{21}	0.01	0.01	0.01	0.01	0.01
	cov_{22}	0.01	0.01	0.01	0.01	0.02
	cov_{23}	0.01	0.01	0.01	0.01	0.01
s_3	cov_{31}	0.01	0.01	0.01	0.01	0.01
	cov_{32}	0.01	0.01	0.01	0.01	0.01
	cov_{33}	0.01	0.01	0.01	0.01	0.01

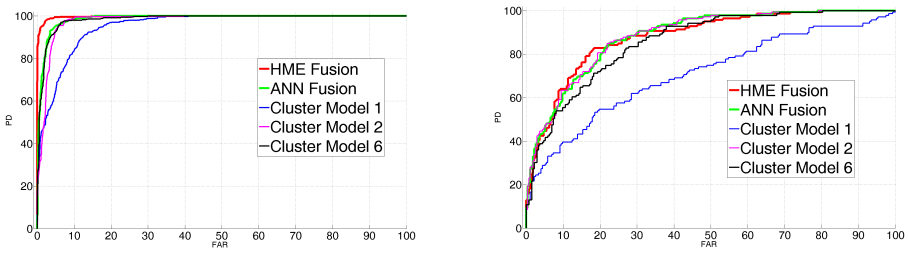
Coefficients

	w
g_{11}	0.42
g_{12}	0.48
g_{13}	0.10
g_{21}	0.46
g_{22}	0.21
g_{23}	0.33
g_{31}	0.10
g_{32}	0.50
g_{33}	0.40

to the training data labels. For each crossvalidation fold, the initial and trained eHMM models, the trained ANN, and the trained HME are saved and will be used to test the respective $\mathfrak{O}_{\mathfrak{l}_{Tst}^{Feat}}$ subset.

5.6.5 eHMM testing

The block diagram for the proposed eHMM in testing mode, was given in figure 22. In the training step, for each crossvalidation fold, we used the $\mathfrak{O}_{\mathfrak{l}_{Trn}^{Feat}}$ subset to learn the eHMM mixture and the decision level fusion model. In this testing step, we use the eHMM models and the decision level component of the eHMM, obtained from the training step, to assign a confidence to a new sequence from the corresponding crossvalidation testing subset $\mathfrak{O}_{\mathfrak{l}_{Tst}^{Feat}}$. For each dataset $\mathfrak{D}_{\mathfrak{l}}$, and experimental setting of discrete vs. continuous and feature extraction method, we repeat the previous step for each crossvalidation fold. The results for a particular experimental setting are summarized in terms of receiver Operating Characteristics (ROC) and Area Under ROC Curve (AUC). In this section, we investigate the performance of the HME and ANN fusion methods for the experimental setting example provided in the eHMM training step. Then, we motivate our choice for the clustering method and the number of clusters in the case of $\mathfrak{D}_3$ using the EHD-based eDHMM. Finally, we



(a) (b)

Figure 67. ROCs generated by the HME fusion, ANN fusion, and few cluster models using (a) the training data ${\mathfrak{O}_3}_{Trn1}^{BAR}$ and, (b) the testing data ${\mathfrak{O}_3}_{Tst1}^{BAR}$.

provide comparative results of the eHMM vs the baseline HMM on all datasets, using the discrete and continuous HMM, and the different feature extraction methods.

5.6.5.1 Comparison of the HME and the ANN fusion results

Continuing with the example used in the eHMM training step, where we have shown the hierarchical clustering results of eCHMM using bar features on ${\mathfrak{O}_3}_{Trn1}^{BAR}$, we reported that the partition of the training data includes four homogeneous clusters and six mixed clusters. In figure 67(a), we show the ROCs generated by the models of one mine-dominated cluster (cluster 6), one clutter-dominated cluster (cluster 1), one mixed cluster (cluster 2) on the training data ${\mathfrak{O}_3}_{Trn1}^{BAR}$. We show also the ROCs of the ANN and HME fusion that were obtained by combining the confidences of all ten cluster models. In figure 67(b), we show the ROCs generated by the same classifiers on the corresponding test data ${\mathfrak{O}_3}_{Tst1}^{BAR}$. As it can be seen in the ROCs of figure 67, the HME and the ANN improve the classification performance over the individual clusters' models with the HME fusion outperfroming the ANN fusion, specially on the training data. In the remainder of our experiments, we use the HME as the default fusion method.

5.6.5.2 Comparison of the clustering methods

In our experiments, we noticed that the clustering methods used, namely the hierarchical agglomerative, the FLeCK, and the spectral FCM-MK, yield comparable partitions in terms of cluster purity (i.e. mine and clutter distributions in each cluster). Therefore, the overall eHMM performance, using one of the clustering algorithms, is comparable in terms of the ROC or the AUC measure. For instance, in figure 68, we show the ROCs generated by the eDHMM on ${\mathfrak{O}_3}^{EHD}$ using the hierarchical, FLeCK, and FCM-MK algorithms with $K = 10$. As it can be seen, the performance

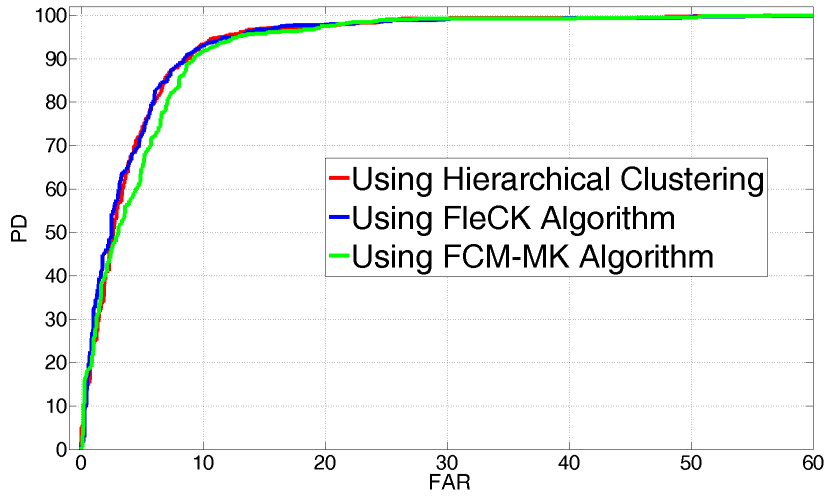


Figure 68. ROCs generated by the eDHMM using ${\mathfrak{O}_3}^{EHD}$ and different clustering algorithms.

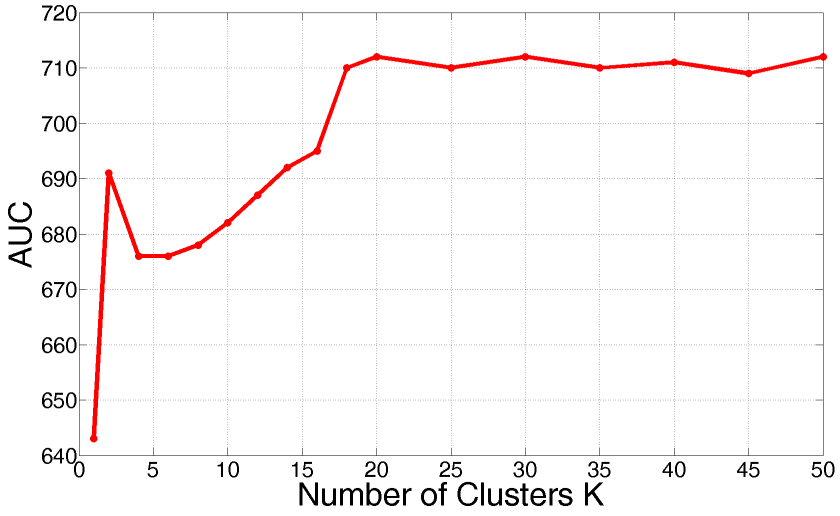


Figure 69. Optimization of the number of clusters

of the eDHMM using one of the clustering methods is comparable. It is worth noting that the FLeCK and FCM-MK algorithms are paramater-based and may have inconsistent results. That is, for the FLeCK algorithm, the results depend on the initial FCM partition. For the FCM-MK, it is not guaranteed that the clustering results in exactly K clusters as the final partition may have empty clusters. Therefore, we will use the hierarchical agglomerative clustering algorithm in the remainder of our experiments, since it is non-parametric and gives consistent results.

5.6.5.3 Choice of the number of clusters K

To optimize the number of clusters, a performance criterion such as the partition purity, the mean of the inter- and intra-cluster distances, and the overall performance of the eHMM can be used. In our case, we use the eHMM's AUC as a criteria for choosing the optimal number of clusters. For that matter, we conducted an empirical study by varying the number of clusters ($K = 1, \cdots, 50$) and measuring the performance of corresponding eHMM AUC for each value of K. In particular, we used the EHD-based eDHMM using the hierarchical clustering and the HME fusion, on $\mathfrak{D}_3$, to illustrate the optimization of the number of clusters. In figure 69, we show the evolution of the eDHMM's AUC with the number of clusters used in the eDHMM hierarchical clustering component. The trivial choice of $K = 1$ is equivalent to the baseline DHMM and has the lowest AUC of 643, as it can be seen in figure 69. For $K = 2$, the eDHMM is comprised of two clusters: one cluster contains the mine signatures and the mine-like clutter signatures, and the other cluster contains the dominantly background, with no significant GPR activity, clutter signatures. Apart from the trivial cases of $K \in \{1, 2\}$, the AUC of the eDHMM increases with number of clusters until it reaches a plateau around $K = 20$. Therefore, we pick $K = 20$ as a reasonable tradeoff between the eDHMM performance and its complexity in terms of number of clusters/ mixture models.

5.6.5.4 eHMM vs. baseline HMM results

In this section, we highlight the relative performance of the eHMM compared to the baseline HMM. For the eHMM, we show the results using the HME fusion and the hierarchical agglomerative clustering with $K = 20$. In figures 70(a)-(d), we show the ROCs generated by the discrete versions of the eHMM and the baseline HMM on dataset $\mathfrak{D}_3$, using one of the feature extraction methods: EHD, Gabor, gradient, and bar respectively. Similarly, in figures 71(a)-(d), we report the ROCs generated by the continuous versions, i.e. the eCHMM and the baseline CHMM, on dataset $\mathfrak{D}_3$, using the EHD, Gabor, gradient, and bar features, respectively. In all the experimental settings using $\mathfrak{D}_3$, the ensemble HMM method outperforms the baseline HMM. In all the ROCs of figures 70 and 71, at a given false alarm rate (FAR), the eHMM has a better probability of detecting targets. For instance, in figure 70(a), at a FAR of 10%, the eDHMM using EHD features successfully identifies 94% of the mines while the baseline DHMM identifies only 87% of the targets. At the same FAR of 10%, the ROCs of figure 71(a) show that the eCHMM successfully identifies 95% while the baseline CHMM probability of detection is 85%.

To gain more insight on the relative performance of the eHMM and the baseline HMM

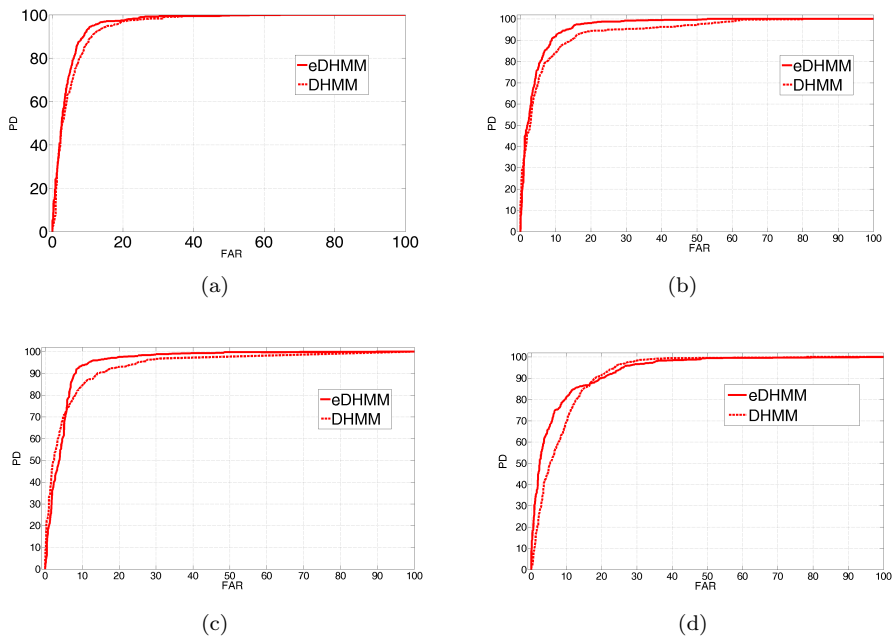


Figure 70. ROCs generated by the eDHMM (solid lines) and baseline DHMM (dashed lines) classifiers using $\mathfrak{D}_3$ and (a) EHD, (b) Gabor, (c) gradient, and (d) bar features

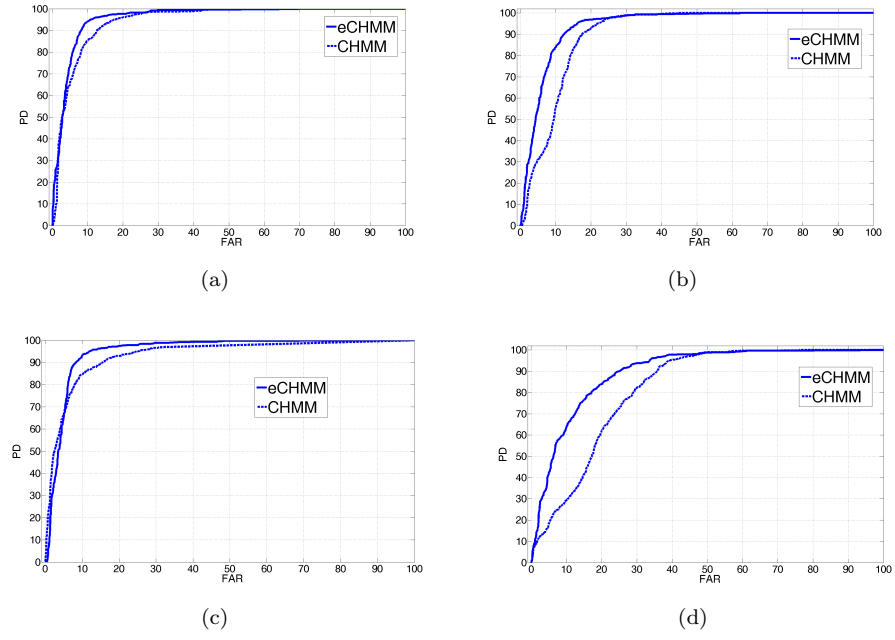


Figure 71. ROCs generated by the eCHMM (solid lines) and baseline CHMM (dashed lines) classifiers using $\mathfrak{D}_3$ and (a) EHD, (b) Gabor, (c) gradient, and (d) bar features

classifiers on $\mathfrak{D}_3$, we plot the scatter-plot of the confidences assigned by the eCHMM vs. the baseline CHMM to the signatures of $\mathfrak{D}_{3Tst}$, in figure 72. The scatter-plot shows that the eCHMM and CHMM classifiers are positively correlated, as most of the points fall along the diagonal of the plot. Both classifiers assign low confidences to clutter and higher confidences to mine signatures specially high metal (in red and purple) and shallow signatures (in red and black). The improvement of the eHMM over the baseline classifier can be seen particularly in the areas R_1 and R_2. In fact, the eCHMM assigns higher confidences to the low-metal signatures enclosed by R_1 while the baseline CHMM assigns relatively lower confidences. Moreover, the eCHMM assigns relatively low confidences ($< .8$) compared to the CHMM (> 6), to the clutter signatures of region R_2.

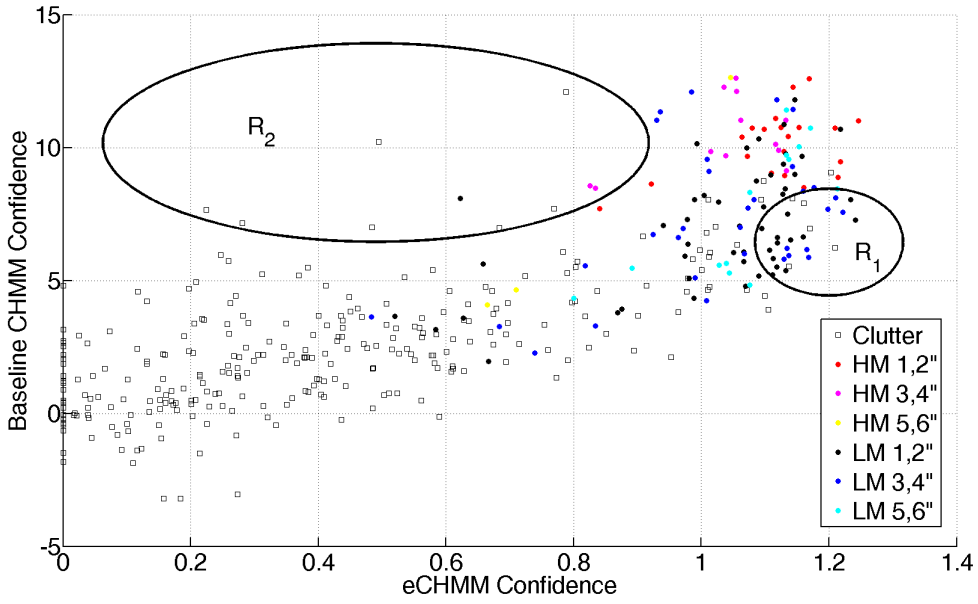


Figure 72. Scatter plot of the confidences assigned by the EHD-based eCHMM (abscissa axis) and the baseline CHMM (ordinate axis) to $\mathfrak{D}_{3Tst1}$ signatures. Clutter, low metal (LM), and high metal (HM) signatures at different depths are shown with different symbols and colors.

The results for all 3 datasets are summarized in terms of and Area Under ROC Curve (AUC) and are reported in table 26. In most of our experiments (24 settings of continuous vs discrete, using EHD, Gabor, gradient, or bar features, for each of the three datasets), the eHMM outperforms the baseline HMM.

TABLE 26

AUC of the eHMM and baseline HMM classifiers

Dataset	Classifier using:	EHD	Gabor	Gradient	Bar
$\mathfrak{D}_1$	eDHMM	**343**	**296**	191	190
	bDHMM	272	122	**299**	**367**
	eCHMM	**326**	**226**	119	**180**
	bCHMM	284	140	**208**	173
$\mathfrak{D}_2$	eDHMM	**402**	**127**	**498**	50
	bDHMM	107	30	400	**314**
	eCHMM	**359**	**122**	**497**	21
	bCHMM	209	102	391	**38**
$\mathfrak{D}_3$	eDHMM	**712**	**719**	**698**	**483**
	bDHMM	643	499	582	471
	eCHMM	**718**	**617**	**697**	**284**
	bCHMM	614	472	579	150

5.7 Chapter summary

In this chapter, the proposed eHMM classifier is applied and evaluated on a real world application: landmine detection. In this application, the eHMM intermediate steps are analyzed and the final performance results are presented. The experiments show that the proposed eHMM intermediate results are inline with the expected behavior and that, overall, the ensemble HMM outperforms the standard HMM. As reported earlier in figures 13, 14, and 15, the clustering step is able to identify similar groups of signatures. The fusion step results for the ANN and the HME method were also presented and analyzed. The results show that both the HME and ANN fusion methods outperform the individual models and that the HME fusion method gives better results than the ANN based fusion. Finally the eHMM versus the baseline HMM ROCs and scatter plots show that the eHMM with both ANN and HME fusion significantly outperforms the baseline two-model classifier.

CHAPTER 6

CONCLUSIONS AND POTENTIAL FUTURE WORK

6.1 Conclusions

In this work, we have proposed a novel ensemble HMM classification method that is based on clustering sequences in the log-likelihood space. The eHMM uses multiple HMM models and fuses them for the final decision making.

We hypothesized that the data are generated by multiple models. These different models reflect the fact that the different classes may have different characteristics accounting for intra-class variability within the data. Our approach has four main steps. First, one HMM is fit to each individual sequence. For each fitted model, we evaluate the log-likelihood of each sequence. This results in a log-likelihood-based similarity matrix. Next, the similarity matrix is partitioned into K groups using a relational clustering algorithm. Three clustering algorithms were evaluated: the hierarchical agglomerative algorithm, the FLeCK algorithm, and the FCM-MK algorithm.

In the third step, we initialize the parameters of one HMM per cluster using the observation sequences belonging to that cluster. Then, we learn the parameters of the HMM models using an optimized training approach for the different K groups depending on their size and homogeneity. For clusters that are dominated by sequences from only one class, we use the standard Baum-Welch re-estimation procedure. For clusters with a mixture of observations belonging to different classes, we use discriminative training based on minimizing the misclassification error to learn a model for each class. Finally, for clusters containing a small number of sequences, we use a variational Bayesian method to update the model parameters given the observed data.

To test a new sequence, its likelihood is computed in all the models and a final confidence value is assigned by combining the multiple models outputs using a decision level fusion method such as an artificial neural network or a hierarchical mixture of experts.

The eHMM, in its discrete and continuous versions, was evaluated on two real-world applications. First, we applied our approach to identify CPR scenes in video simulating medical crises. In this application, we used a small dataset of 2-dimensional CPR and non-CPR sequences that

represent the motion vectors extracted from the simulation videos. Therefore, we illustrated and visualized the intermediate steps of the eHMM and compared its performance to the baseline HMM. In the second application, we evaluated the eHMM on the task of landmine detection using GPR data and multiple feature representations. In particular, we applied our method on three large datasets of GPR data using EHD, Gabor, gradient, and bar features. In this application, we evaluated the individual components of the eHMM and emphasized on its overall performance compared to the baseline HMM.

Results on the CPR data and on the large GPR data collections show that the proposed method can identify meaningful and coherent HMM clusters models that describe different properties of the data. Each HMM mixture component models a group of data that share common attributes. The experiments show that the proposed eHMM intermediate results are inline with the expected behavior. The results indicate that, in both the continuous and discrete versions, the proposed method outperforms the baseline HMM that uses one model for each class in the data.

6.2 Potential future work

In the application of the eHMM to the GPR data, we used each feature extraction separately to build the eHMM mixture models. Therefore, multiple models can be learned from the different features. We suggest combining each of the single-feature-based eHMMs either at the similarity matrix level, or at the fusion level. At the similarity matrix level, we can combine the similarity matrices of each single-feature-based eHMM and produce a weighted-average similarity matrix that will be used in the relational clustering. At the fusion level, we can adjust the ANN or HME architecture to take the mixture models of all the single-feature-based eHMMs as input. Subsequently, the fusion model parameters will be learned from the training data. Finally, we can invoke a relational clustering method that simultaneously clusters the training data using several similarity matrices and assigns a weight to each similarity matrix in each cluster. Those weights can be used at the decision level fusion to decide which single-feature-based model will be used for each cluster.

Future work also includes the evaluation of the eHMM with other applications such as face recognition, gene sequence alignment, and handwritten character recognition.

REFERENCES

[1] Dar-Shyang Lee and S. N. Srihari, "A theory of classifier combination: the neural network approach," in *ICDAR '95: Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1)*, Washington, DC, USA, 1995, p. 42, IEEE Computer Society.

[2] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, 2000.

[3] K. J. Hintz, "Snr improvements in niitek ground penetrating radar," *in Proceedings of the SPIE Conference on Detection and Remediation Technologies for Mines and Minelike Targets IX, Orlando, FL, USA, April*, 2004.

[4] L. R. Rabiner and B. H. Juang, "An introduction to hidden markov models," *IEEE ASSP Magazine*, pp. 4–16, 1986.

[5] Anders Krogh, Michael Brown, I. Saira Mian, Kimmen Sjlander, and David Haussler, "Hidden markov models in computational biology : Applications to protein modeling," *Journal of Molecular Biology*, vol. 235, no. 5, pp. 1501 – 1531, 1994.

[6] Chris Burge and Samuel Karlin, "Prediction of complete gene structures in human genomic dna," *J. Mol. Biol*, vol. 268, pp. 78–94, 1997.

[7] Lawrence R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, 1989, pp. 257–286.

[8] G Forney, "The Viterbi algorithm," *Proc. IEEE, vol. 61, pp. 268-278*, 1973.

[9] Leonard E. Baum and Ted Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The Annals of Mathematical Statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[10] L. Le Cam, "Maximum likelihood: An introduction," *Intl. Stat. Rev*, vol. 58, pp. 153–171, 1990.

[11] Biing-Hwang Juang, Wu Chou, and Chin-Hui Lee, "Minimum Classification Error Rate Methods for Speech Recognition," *Transactions on Speech and Audio Processing, VOL. 5, No.3*, May 1997.

[12] L Bahl, P Brown, P De Souza, and R Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition,," *in Proc. ICASSP 86,pp. 4952.*, 1986.

[13] Robert E. Schapire, "A brief introduction to boosting," in *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, San Francisco, CA, USA, 1999, pp. 1401–1406, Morgan Kaufmann Publishers Inc.

[14] Leo Breiman, "Bagging predictors," in *Machine Learning*, 1996, pp. 123–140.

[15] David H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241 – 259, 1992.

[16] Tin Kam Ho, "The random subspace method for constructing decision forests," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 8, pp. 832 –844, aug 1998.

[17] L.I. Kuncheva, "A theoretical study on six classifier fusion strategies," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 2, pp. 281 –286, feb 2002.

[18] Hunkapillier T McClure M Baldi P, Chauvin Y, "Hidden markov models of biological primary sequence information.," *Proc. Nat. Acad. Sci. USA 91: 1059-1063*, 1994.

[19] Grant G R Ewens W J, *Statistical methods in bioinformatics: An introduction.*, Springer-Verlag, New York, 2001.

[20] Dorffner G Tino P, Schittenkopf C, "Temporal pattern recognition in noisy non-stationary time series based on quantization into symbolic streams: Lessons learned from financial volatility trading.," 2000.

[21] Shreeya Sengupta, Hui Wang, William Blackburn, and Piyush Ojha, "Financial sequences and the hidden markov model," in *Global Trends in Information Systems and Software Applications*, P.Venkata Krishna, M.Rajasekhara Babu, and Ezendu Ariwa, Eds., vol. 270 of *Communications in Computer and Information Science*, pp. 5–12. Springer Berlin Heidelberg, 2012.

[22] Olivier Cappé, Eric Moulines, and Tobias Ryden, *Inference in Hidden Markov Models (Springer Series in Statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[23] J Hartigan, *Clustering Algorithms*, Wiley, 1975.

[24] R Duda and P Hart, *Pattern Classification and Scene Analysis.*, John Wiley and Sons, 1973.

[25] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*, Cambrdige University Press, 2003.

[26] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[27] Christophe Andrieu, "An introduction to MCMC for machine learning," 2003.

[28] Alan E. Gelfand and Adrian F. M. Smith, "Sampling-based approaches to calculating marginal densities," *Journal of the American Statistical Association*, vol. 85, no. 410, pp. 398–409, 1990.

[29] David J.C. MacKay, "Ensemble learning for hidden markov models," Tech. Rep., 1997.

[30] Oualid Missaoui, *Generalized Multi-Stream Hidden Markov Models*, Ph.D. thesis, University of Louisville, Louisville, Kentucky, 2010.

[31] Y Zhao, Paul Gader, P Chen, and Y Zhang, "Training DHMMs of mine and clutter to minimize landmine detection errors.," *IEEE Trans. Geosci. Remote Sensing*, 2003.

[32] D. Chandler, *Introduction to Modern Statistical Mechanics*, Oxford University Press, 1987.

[33] Hichem Frigui, Oualid Missaoui, and Paul Gader, "Landmine detection using discrete hidden Markov models with Gabor features," *in SPIE Conf. Detection and Remediation Technologies for Mines and Minelike Targets XII, April*, 2007.

[34] Ángel de la Torre, Antonio M. Peinado, Antonio J. Rubio, José C. Segura, and Carmen Benítez, "Discriminative feature weighting for hmm-based continuous speech recognizers," *Speech Commun.*, vol. 38, no. 3-4, pp. 267–286, 2002.

[35] C Chibelishi, J Mason, and F Deravi, "Feature Level Data Fusion For Bimodal Person Recognition," *International Conference on Image Processing and Its Applications (ICIP)*, 1997.

[36] V Chatziz, A Bors, and I Pitas, "Multimodal decision level fusion for person authentication," *IEEE Trans. Syst. Man Cybern. A 29*, 1999.

[37] Zoubin Ghahramani and Michael Jordan, "Factorial Hidden Markov Models," *Neural Information Processing Systems (NIPS)*, 1995.

[38] A Nefian, L Liang, T Fu, and X Liu, "A Bayesian approach to audio-visual speaker identification," *Fourth International Conf. Audio- and Video-based Biometric Person Authentication*, 2003.

[39] Potamianos Gerasimos, Neti Chalapathy, Luettin Juergen, and Matthews Iain, "Audio-Visual Automatic Speech Recognition: An Overview," *Audio-Visual Speech Processing, E. Vatikiotis-Bateson, G. Bailly, and P. Perrier (Eds.), MIT Press, ISBN: 0-26-222078-4*, 2009.

[40] A.K. Jain, R.P.W. Duin, and Jianchang Mao, "Statistical pattern recognition: a review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 1, pp. 4 –37, jan 2000.

[41] Leo Breiman, "Prediction games and arcing algorithms," *Neural Comput.*, vol. 11, no. 7, pp. 1493–1517, 1999.

[42] Vladimir N. Vapnik, *The nature of statistical learning theory*, Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[43] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," 1997.

[44] Thomas G. Dietterich, "Ensemble methods in machine learning," in *MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems*, London, UK, 2000, pp. 1–15, Springer-Verlag.

[45] M.I. Jordan, "Hierarchical mixtures of experts and the em algorithm," in *Advances in Neural Networks for Control and Systems, IEE Colloquium on*, 25-27 1994, pp. 1/1 –1/3.

[46] Michael I. Jordan and Robert A. Jacobs, "Hierarchies of adaptive experts," in *NIPS*, 1991, pp. 985–992.

[47] H. Frigui, Lijun Zhang, and P. Gader, "Context-dependent multi-sensor fusion for landmine detection," in *Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*, 7-11 2008, vol. 2, pp. II–371 –II–374.

[48] A. Abdallah, H. Frigui, and P. Gader, "Context extraction for local fusion using fuzzy clustering and feature discrimination," in *Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on*, 20-24 2009, pp. 490 –495.

[49] V.I. Gorodetskiy and S.V. Serebryakov, "Methods and algorithms of collective recognition," *Automation and Remote Control*, vol. 69, no. 11, pp. 1821–1851, 2008.

[50] R.A. Jacobs, "Methods for combining experts probability assessments," *Neural Computation*, vol. 7, no. 5, pp. 867–888, 1995.

[51] Esko Ukkonen, "On approximate string matching," in *Foundations of Computation Theory*, Marek Karpinski, Ed., vol. 158 of *Lecture Notes in Computer Science*, pp. 487–495. Springer Berlin Heidelberg, 1983.

[52] O. Bchir and H. Frigui, "Fuzzy clustering with learnable cluster dependent kernels," in *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, June 2011, pp. 2521–2527.

[53] N. Baili and H. Frigui, "Fuzzy clustering with multiple kernels in feature space," in *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*, June 2012, pp. 1–8.

[54] J. H. Ward, "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 236–244, 1963.

[55] Sergios Theodoridis and Konstantinos Koutroumbas, *Pattern Recognition, Third Edition*, Academic Press, Inc., Orlando, FL, USA, 2006.

[56] J.W. Davenport R.J. Hathaway and J.C. Bezdek, "Relational duals of the c-means algorithms," in *Pattern Recognition*, 1989, vol. 22, p. 205.

[57] J.C. Bezdec, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

[58] J.C. Bezdek and R.J. Hathaway, "Vat: a tool for visual assessment of (cluster) tendency," in *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, 2002, vol. 3, pp. 2225–2230.

[59] John H. Holland, *Adaptation in natural and artificial systems*, MIT Press, Cambridge, MA, USA, 1992.

[60] S. Kirkpatrick, Jr. Gelatt, C. D., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[61] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[62] James C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[63] Giorgio Fumera and Fabio Roli, "A theoretical and experimental analysis of linear combiners for multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 942–956, 2005.

[64] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," pp. 318–362, 1986.

[65] Stephen José Hanson and Lorien Pratt, "Comparing biases for minimal network construction with back-propagation," pp. 177–185, 1989.

[66] Zhang H. J. and Wu J. et al., "An integrated system for content-based video retrieval and browsing," *Pattern Recognition*, vol. 30, no. 4, pp. 643–658, 1997.

[67] Yeung M. M. and Liu B., "Efficient matching and clustering of video shots," in *IEEE International Conference on Image Processing*, 1995.

[68] Zhuang Y. and Y. Rui et al., "Adaptive key frame extraction using unsupervised clustering," in *IEEE International Conference on Image Processing*, 1998.

[69] Lagendijk R. L. and Hanjalic A. et al., "Visual search in a smash system," in *IEEE International Conference on Image Processing*, 1996.

[70] Shahraray B. and Gibbon D. C., "Automatic generation of pictorial transcriptions of video programs," in *IS&T/SPIE International Conf.*, 1995, pp. 512–519.

[71] Ju S. X. and Black M. J. et al., "Summarization of videotaped presentations: automatic analysis ofmotion and gesture," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 686–696, 1998.

[72] Taniguchi Y. and Akutsu A. et al., "An intuitive and efficient access interface to real-time incoming video based on automatic indexing," in *ACM international conference on Multimedia*, 1995, pp. 25–33.

[73] Smoliar S. W. and Zhang H. J., "Content based video indexing and retrieval," *IEEE Transactions on Multimedia*, vol. 1, no. 2, pp. 62–72, 1994.

[74] C.O. Conaire, N.E. O'Connor, and A.F. Smeaton, "Detector adaptation by maximising agreement between independent data sources," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, June 2007, pp. 1–6.

[75] O. Javed A. Yilmaz and M. Shah, "Object tracking: A survey," *ACM Computing Surveys (CSUR)*, vol. 38, 2006.

[76] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Inteligence*, vol. 17, 1981.

[77] "International campaign to ban landmines, landmiane monitor report 2013 (ottawa: Mines action canada: October 2013), p. 37," .

[78] T.R. Witten, "Present state of the art in ground-penetrating radars for mine detection," *in SPIE Conf Detection and Remediation Technologies for Mines and Minelike Targets III, orlando, FL, pp. 576-586*, 1998.

[79] B. Nelson G. Vaillette P.D. Gader, H. Frigui and J.M. Kelle, "New results in fuzzy set based detection of landmines with gpr," *in SPIE Conf Detection and Remediation Technologies for Mines and Minelike Targets IV, orlando, FL, pp. 1075-1084*, 1999.

[80] D. Carevic, "Kalman filter-based approach to target detection and target-background separation in ground-penetrating radar data," *in SPIE Conf Detection and Remediation Technologies for Mines and Minelike Targets IV, orlando, FL, pp. 1284-1288*, 1999.

[81] A. Gunatilaka and B.A. Baertlein, "Subspace decomposition technique to improve gpr imaging of anti-personal mines," *in SPIE Conf Detection and Remediation Technologies for Mines and Minelike Targets V, orlando, FL*, 2000.

[82] H. Brunzell, "Detection of shallowly burried objects using impulse radar," *IEEE Trans. Geoscience and Remote Sensing, vol. 37, pp. 875-886*, 1999.

[83] R.K. Mehra S. Yu and T.R. Witten, "Automatic mine detection based on ground penetrating radar.," *in SPIE Conf Detection and Remediation Technologies for Mines and Minelike Targets IV, orlando, FL, pp. 961-972*, 1999.

[84] K. Satyanarayana H. Frigui and P. Gader, "Detection of landmines using fuzzy and possibilistic membership functions.," *in proceedings of the IEEE Conference on Fuzzy Systems, Saint Louis, Missouri*, 2003.

[85] D. Carevic, "Clutter reduction and target detection in ground penetrating radar using wavelets.," *in Proceedings of the SPIE Conference on Detection and Remediation Technologies for Mines and Minelike Targets IV, Orlando, FL, USA*, 1999.

[86] M. Mystkowski P. Gader and Y. Zhao, "Landmine detection with ground penetrating radar using hidden markov models.," *IEEE Trans, Geoscience and Remote Sensing, vol.39, pp. 1231-1244*, 2001.

[87] V. S. Munshi S. L. Tantum, Y. Wei and L. M. Collins, "A comparison of algorithms for landmine detection and discrimination using ground penetrating radar.," *in Proceedings of the SPIE Conference on Detection and Remediation Technologies for Mines and Mineslike Targets, Orlando, FL, USA*, 2002.

[88] Paul Gader Hichem Frigui, K.C.Ho, "Real-time landmine detection with ground-penetrating radar using discriminative and adaptive hidden markov models.," *EURASIP Journal on Applied Signal Processing 2005:12, 1867-1885.*

[89] H. Frigui G. Vaillette P. Gader, B. Nelson and J. Keller, "Fuzzy logic detection of landmines with ground penetrating radar.," *Signal Processing, special issue on fuzzy logic in signal processing, vol. 80, pp. 1069-1084,*, 2000.

[90] W. H. Lee P. Gader and J. N. Wilson, "Detecting landmines with ground penetrating radar using feature-based rules, order statistics, and adaptive whitening,," *IEEE Trans, Geoscience and Remote Sensing, vol. 42, no. 11, pp.2522-2534,*, 2004.

[91] Gader P. Frigui H. and Kotturu S., "Detection and discrimination of landmines in ground penetrating radar using an eigenmine and fuzzy membership function approach," *in SPIE Conf. Detection and Remediation Technologies for Mines and Minelike Targets, April*, 2004.

[92] H. Frigui and P. D. Gader, "Detection and discrimination of land mines based on edge histogram descriptors and fuzzy k-nearest neighbors," *in Proceedings of the IEEE International Conference on Fuzzy Systems, Vancouver, BC, Canada, July*, 2006.

[93] C. S. Throckmorton P. A. Torrione and L. M. Collins, "Performance of an adaptive feature-based processor for a wideband ground penetrating radar system,," *IEEE Trans. Aerospace and Electronic Systems (in press).*

[94] H. Frigui and P. Gader, "Detection and discrimination of landmines in ground-penetrating radar based on edge histogram descriptors," *in Proc.SPIE Conf. Detect. Remediation Technol. Mines Minelike Targets, pp. 62176233.*, 2006.

[95] B. S. Manjunath, P. Salembier, and T. Sikora, *Introduction to MPEG-7: Multimedia Content Description Interface.*, WILEY, July ,2002.

[96] P. Gader, Magdi Mohamed, and Jung-Hsien Chiang, "Comparison of crisp and fuzzy character neural networks in handwritten word recognition," *Fuzzy Systems, IEEE Transactions on*, vol. 3, no. 3, pp. 357 –363, aug 1995.

CURRICULUM VITAE

NAME: Anis Hamdi

ADDRESS: Department of Computer Engineering and Computer Science
Speed School of Engineering
University of Louisville
Louisville, KY 40292

EDUCATION:

Ph.D., Computer Science and Engineering
December 2014
University of Louisville, *Louisville, Kentucky*

B.E., Signals and Systems
June 2002
Polytechnic School of Tunisia, *Tunis, Tunisia*

CONFERENCE PUBLICATIONS:

1. **A. Hamdi** and H. Frigui. *"Evaluation of various feature extraction methods for landmine detection using hidden Markov models"*, SPIE Conf. Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XVII, Volume 8357, May 2012.

2. **A. Hamdi**, O. Missaoui, and H. Frigui. *"An SVM classifier with HMM-based kernel for landmine detection using ground penetrating radar"*, Geoscience and Remote Sensing Symposium (IGARSS), 2010 IEEE International, pages 4196−4199, July 2010.

3. **A. Hamdi**, O. Missaoui, and H. Frigui. *"Landmine detection using ensemble discrete hidden Markov models with context dependent training methods"*, SPIE Conf. Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XV, Volume 7664, April 2010.

4. **A. Hamdi** and H. Frigui. *"Landmine detection using an ensemble of continuous HMMs with multiple features"*, Geoscience and Remote Sensing Symposium (IGARSS), 2011 IEEE International, pages 63−66, July 2011.

5. H. Figui, **A. Hamdi**, O. Missaoui, and P. Gader. *"Landmine detection using mixture of discrete hidden markov models"*, SPIE Conf. Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XIV, Volume 7303, May 2009.

6. H. Frigui, S. Rawungyot, and **A. Hamdi**, *"Identification of Cardio Pulmonary Resuscitation (CPR) scenes in video simulating medical crises"*, International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Sousse, Tunisia, March 2014.